

2

NAVAL POSTGRADUATE SCHOOL MONTEREY, CALIFORNIA

AD-A279 307



THESIS

DTIC
ELECTE
MAY 17, 1994
S B D

**The Communications Toolbox for MATLAB
and
E0 3513 Laboratory Design**

by

Susan A. Guckelberg

March 1994

**Thesis Advisor:
Second Reader:**

**Randy I. Borchardt
Dan C. Boger**

Approved for public release; distribution is unlimited

94-14747

94 5 17 023

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
<small>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</small>				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE 24 March 1994		3. REPORT TYPE AND DATES COVERED final
4. TITLE AND SUBTITLE The Communications Toolbox for MATLAB and EO 3513 Laboratory Design			5. FUNDING NUMBERS	
6. AUTHOR(S) Guckelberg, Susan A.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT unclassified/unlimited			12b. DISTRIBUTION CODE A	
13. ABSTRACT (Maximum 200 words) <p>EO 3513, <u>Communications Systems Engineering II: Modulation</u>, is the second of a three-course sequence for students in the C3, Space Systems Operations, and Information Technology Management curricula at the Naval Postgraduate School in Monterey, California. This course presents a review of Fourier methods and covers analog and digital communications systems.</p> <p>The identified need for computer laboratories to support EO 3513 results in the development of a set of 34 functions collectively called the Communications Toolbox for use with MATLAB. The Communications Toolbox contains functions that, when linked together, simulate the output of various communications systems.</p> <p>Developed in association with the Communications Toolbox are two sets of laboratories: nine computer-aided laboratories (tutorial in nature), and fourteen programming laboratories.</p> <p>Laboratory and toolbox development are described and documented, with additional notes on design, testing, and implementation. The complete laboratory sets, with answer keys, User's Guide, and computer code for toolbox functions, are provided.</p>				
14. SUBJECT TERMS digital encoding, electronic communications, modulation, quantization, sampling			15. NUMBER OF PAGES 479	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT unclassified	20. LIMITATION OF ABSTRACT UL	

Approved for public release; distribution is unlimited.

THE COMMUNICATIONS TOOLBOX FOR MATLAB

AND

EO 3513 LABORATORY DESIGN

by

Susan A. Guckelberg
Lieutenant, United States Navy
B.A., Luther College, 1976

Submitted in partial fulfillment of the requirements for
the degree of

MASTER OF SCIENCE IN

INFORMATION TECHNOLOGY MANAGEMENT

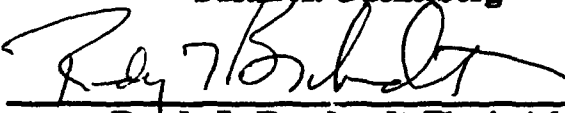
from the

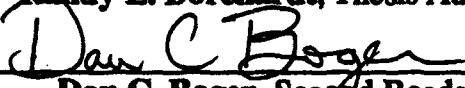
NAVAL POSTGRADUATE SCHOOL
March 1994

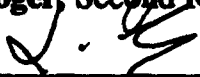
Author:


Susan A. Guckelberg

Approved by:


Randy L. Borchardt, Thesis Advisor


Dan C. Boger, Second Reader


David R. Whipple, Chairman, Department of
Systems Management

ABSTRACT

EO 3513, Communications Systems Engineering II: Modulation, is the second of a three-course sequence for students in the C3, Space Systems Operations, and Information Technology Management curricula at the Naval Postgraduate School in Monterey, California. This course presents a review of Fourier methods and covers analog and digital communications systems.

The identified need for computer laboratories to support EO 3513 results in the development of a set of 34 functions collectively called the Communications Toolbox for use with MATLAB. The Communications Toolbox contains functions that, when linked together, simulate the output of various communications systems.

Developed in association with the Communications Toolbox are two sets of laboratories: nine computer-aided laboratories (tutorial in nature), and fourteen programming laboratories.

Laboratory and toolbox development are described and documented, with additional notes on design, testing, and implementation. The complete laboratory sets, with answer keys, User's Guide, and computer code for toolbox functions, are provided.

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

TABLE OF CONTENTS

I	INTRODUCTION AND BACKGROUND.....	1
II	DESIGN NOTES.....	3
	A. GENERAL LABORATORY CONCEPT	3
	B. INITIAL LABORATORY OBJECTIVES.....	4
III	DEVELOPMENT NOTES.....	6
	A. A PARADIGM SHIFT	7
	B. DEVIATION FROM OBJECTIVES	7
IV.	IMPLEMENTATION AND TESTING NOTES	8
	A. TOO LITTLE, TOO LATE	8
	B. FEEDBACK VIA STUDENT CRITIQUES	8
	C. INSTRUCTOR PREFERENCES	9
	D. INCORPORATION OF FEEDBACK	10
	E. INSTRUCTOR-FRIENDLY FORMAT	10
	F. REVISED OBJECTIVES AND REFERENCES	11
V.	CONCLUSION.....	20
	LIST OF REFERENCES	21
	APPENDIX A (COMPUTER-AIDED LABORATORIES).....	23
	APPENDIX B (COMPUTER-AIDED LABORATORY KEYS).....	79
	APPENDIX C (PROGRAMMING LABORATORIES).....	213
	APPENDIX D (PROGRAMMING LABORATORY KEYS).....	285
	APPENDIX E (COMMUNICATIONS TOOLBOX FOR MATLAB)	407
	APPENDIX F (USER'S GUIDE TO THE COMMUNICATIONS TOOLBOX FOR MATLAB)	447

APPENDIX G (COMPARISON OF REQUIREMENTS).....471
INITIAL DISTRIBUTION LIST.....472

**This page is intentionally
left blank.**

I. INTRODUCTION AND BACKGROUND

EO 3513. Communications Systems Engineering II: Modulation is the second of a redesigned three-course sequence for the C3, Space Systems Operations, and Information Technology Management curricula at the Naval Postgraduate School in Monterey, California. The EO sequence provides students with the background required to understand and apply basic telecommunications and computer technology principles in operational environments.

The content of the previous courses came under review by the C3 Academic Group as part of a strategy to address the concerns highlighted in the 1992 curriculum review [Moose 93]. Problems addressed by the group included the lack of "homogeneity" in student academic background and a need to extend the basics of the technological support systems—communications systems, computers, and sensors—to cover the engineering aspects of C3 systems. (For example, students should understand why specific systems designs are chosen and know the limitations of various technologies that may hamper development of future systems.)

Among the specific actions recommended were the adoption of a new course matrix in the communications and sensors area and the raising of the required student Academic Profile Code to reflect the completion of a calculus-based physics course. The new matrix eliminated MA 1248, Selected Topics for Applied Mathematics, which introduced Fourier series and Fourier integral transforms.

Instead, these topics, along with complex mathematics and an introduction to circuits, are taught in the first course of the new EO sequence, EO 2413. EO 3513 follows with a review of Fourier analysis, analog modulation, sampling theorem, spectral representation of pulse and digital systems, pulse and digital modulations, baseband coding forms, and frequency- and time-division multiplexing. The final course of the sequence, EO 3503, includes the effects and measurement of noise models, channel capacity and antennas, other transmission media, error correction, the performance of complete systems, and an introduction to communications security.

The development of this unique course sequence indicated the necessity for the concurrent development of specialized supporting laboratories. It was the opinion of the Curriculum Working Group that computer-based laboratories allowing students to program inputs and analyze outputs would best suit the objectives of EO 3513. MATLAB, a software package developed by The MathWorks, Inc., was felt to be the optimal programming environment for these laboratories.

This document chronicles the development and implementation effort for the laboratories supporting EO 3513 as initiated by the Curriculum Working Group. The contents of the Communications Toolbox for use with MATLAB and two associated laboratory sets are included as appendices.

II. DESIGN NOTES

A. GENERAL LABORATORY CONCEPT

The predominant concept for a set of laboratories supporting EO 3513 was that students would be able to work with a set of MATLAB functions (".m" files) that would perform modulation, demodulation, radio frequency conversion, and similar procedures. By linking the functions together, students could simulate the outputs of communications systems introduced in the classroom. The collective functions would be known as the Communication Toolbox. This concept had the additional benefit of reinforcing computer programming skills and theoretical concepts without requiring proficiency in using laboratory equipment.

It was proposed that the laboratories be written during the Summer 1993 "development" quarter while a session of EO 2750 (precursor to EO 3513) was being taught. Laboratories and newly-developed MATLAB function files would be provided to students each week to support classroom instruction. The following "clean-up" quarter (Fall 1994), during which EO 3513 would not be offered, would provide an opportunity to fine-tune the laboratories and Communications Toolbox before the course was taught again in the Winter 1994 quarter.

Both advantages and disadvantages were present in this proposal. A truly concentrated effort would be required to generate and test the code needed

during the 11-week quarter. However, feedback would be almost immediate, and with a large course enrollment (more than 100 students), functional testing would be brutally thorough. The development challenge was accepted.

B. INITIAL LABORATORY OBJECTIVES

In June 1993, the following objectives were provided for the laboratory set:

- **Laboratory 1 -Fourier Techniques Review**

- Evaluate the spectrum of a signal using Fourier methods

- Evaluate the effect of filters on a signal

- Differentiate between time and frequency domain representations of a signal

- **Laboratory 2 -Sampling and Reconstruction**

- Observe the effect of sampling on spectrum

- Observe the effect of aliasing

- Observe the different types of reconstruction

- **Laboratory 3 -PAM, PWM, PPM**

- Observe difference in the time and frequency domain between the original signal and its pulse representation

- **Laboratory 4 -Analog to Digital Conversion and Digital Encoding**

- Observe the effect of quantization noise

- Observe the process of PCM encoding

- Calculate the dynamic range of a system

- Observe the effect of companding

- **Laboratory 5 -AM SSB and AM DSB**

Observe the time and frequency domains for both types of modulation with inputs: (1) single tone, (2) two tones, (3) simulated speech

Calculate the power and bandwidth of the above signals

- **Laboratory 6 - Conventional AM**

Observe the time and frequency domain for conventional AM modulation with inputs: (1) single tone, (2) two tones, (3) simulated speech

Calculate the power and bandwidth of the above signals

- **Laboratory 7 - Frequency Modulation and Angle Modulation**

Observe the time and frequency domain for both with inputs: (1) single tone, (2) two tones, (3) simulated speech

Calculate the power and bandwidth of the above signals

- **Laboratory 8 - Frequency and Time Division Multiplexing**

Observe the processes in the time and frequency domains

Be able to explain in words what each is doing

- **Laboratory 9 - RF Digital Modulation: ASK, FSK, DPSK, QPSK**

Observe the processes in the time and frequency domains

Be able to explain in words what each is doing

III. DEVELOPMENT NOTES

An initial set of six of the nine proposed laboratories was developed during the Summer 1993 quarter. The remaining laboratories were disposed of as follows:

- Laboratory 1, Fourier Techniques Review, was a topic that the course instructors felt they could best develop themselves.
- Laboratory 6, Conventional AM, was merged into Laboratory 5, AM SSB and AM DSB, due to the similarity of the subject matter.
- Laboratory 8, Frequency and Time Division Multiplexing, was dropped. The topics were such that instructors "wove" them into their lectures wherever relevant. There appeared to be no optimal placement for a laboratory on multiplexing.

The majority of development effort during the Summer 1993 quarter was spent writing code for the functions and ensuring their interoperability. The primary focus was on functionality rather than performance, partly due to the lack of leisure time enforced by the schedule—and the knowledge that there would be plenty of time to improve the functions during the cleanup quarter.

Following limited testing of the functions by the developer, the associated laboratory was drafted, then approved by one or more of the instructors. A narrative format was chosen because it was explanatory rather than cryptic.

Answer keys and sample scripts were not specifically provided until Laboratory 4. At that time, an appreciation was gained for the amount of time

invested by students in performing calculations and printing plots, in addition to the primary task of coding the function calls and producing plots.

A. A PARADIGM SHIFT

With the development of Laboratory 5 came a shift in the laboratory focus. The developer was requested by instructors to provide a MATLAB script file for students to run, rather than require students to write any MATLAB code. Subsequently, Laboratories 5, 7, and 9 were developed with a tutorial aspect absent in Laboratories 2, 3, and 4.

B. DEVIATION FROM OBJECTIVES

The initial objectives were altered in the following cases:

- Laboratory 2 used a signal recovery method in the frequency domain (*filtering*) rather than in the time domain (*reconstruction*).
- Laboratories 5 and 7 employed three-tone rather than two-tone signals for the multi-tone input.
- Laboratories 5 and 7 did not incorporate a simulated speech input.
- Laboratory 7 focused only on frequency modulation. Because of the similarity of frequency and phase modulation, phase modulation is not widely covered.
- Laboratory 9 replaced differential phase-shift keying (DPSK) with binary phase shift keying (BPSK) as one of four digital modulation methods explored.

IV. IMPLEMENTATION AND TESTING NOTES

A. TOO LITTLE, TOO LATE

The attempt at simultaneous laboratory development and implementation proved in the end to be unresponsive to the pulse of the classroom. Ideally, a student is given a laboratory handout at the time its topic is introduced in the classroom. The instructor will have had the opportunity to review the laboratory content before finalizing lecture notes, so that lecture and laboratory reinforce one another. The ambitious development schedule for the weekly laboratories was simply too tight to allow adequate review by instructors prior to distribution to students.

Laboratories 7 and 9 were each completed on schedule, but a few days too late to do little more than increase student/instructor workload as the classroom lecture pace quickened towards the end of the quarter. Therefore only Laboratories 2, 3, 4, and 5 were assigned to students.

B. FEEDBACK VIA STUDENT CRITIQUES

A feedback sheet was attached to each laboratory handout. Approximately 120 students were enrolled in five sections of EO 2750 during the Summer 1993 quarter. While it is not known exactly how many laboratories were assigned and

with what direction, approximately 129 student feedback forms were returned to the developer. While this number provided ample student feedback, the feedback tended to focus on problems created by the large enrollment: overcrowding of the laboratories, competition for printers, corruption of files, and other trying circumstances.

Discounting specific problems with Toolbox functions and computer facility resources, feedback generally fell into one of five areas:

- "Teach me EO, not MATLAB!" (Emphasize analysis, not programming.)
- The laboratories require too much memory to run on the average personal computer.
- The laboratories take too much time to complete.
- Questions on the laboratories are too vague.
- There is no opportunity to vary the input parameters and see the effects on the signals.

C. INSTRUCTOR PREFERENCES

It became apparent rather quickly that philosophies among the three instructors regarding use of the laboratories differed widely. Laboratories were assigned in full, in part, or not at all. They were assigned to individuals or to groups. In one case, a set of graphs was produced and duplicated for others to use, so that computer time was avoided altogether.

Nonetheless, the developer remained determined to produce a laboratory set that would simultaneously satisfy several instructors' needs.

D. INCORPORATION OF FEEDBACK

While the programming-type laboratories were abandoned fairly early during the development quarter, their value as a learning vehicle was never questioned. The developer chose to promote the harmonious existence of two independent, related sets of laboratories: "computer-aided" laboratories that require no programming whatsoever, and "programming" laboratories, smaller in scope, with clearly defined procedures designed not to frustrate students.

Other changes stimulated by student feedback include the following:

- Laboratories are considerably shortened (sometimes by dividing into smaller separate laboratories).
- Laboratory 1, Signal and Spectrum Generation, assumes no prior experience with MATLAB.
- To better manage memory, all scripts to be run by students were developed on a platform with just 4 megabytes of memory installed. Programming laboratories include instructions to help manage memory.
- The wording of all questions is clarified significantly, with all necessary formulas provided in the laboratory handout.
- Plots are kept to a minimum by overlaying signals whenever possible. Gratuitous labeling of plots is avoided.
- In programming laboratories, MATLAB coding requirements appear in bold type and are preceded either by the words "M-file" or "Plot."

E. INSTRUCTOR-FRIENDLY FORMAT

When an assignment is difficult to correct, its return to students is invariably delayed. Instructor needs are taken into account in the following ways:

- Recognizing that instructors may ignore certain topics in favor of emphasizing others, a modular structure is adopted. For

example, Computer-aided Laboratories 5-8, which deal with single- and multi-tone input, delay the treatment of multi-tone input so that it can be easily eliminated from the assignment. In Computer-aided Laboratory 9, each RF digital modulation section can be run independently.

- Questions and plots are numbered so that instructors can easily choose from among them.
- Answer keys contain (1) a complete set of questions and answers, (2) a complete set of labeled plots, and (3) MATLAB code for either the "scr.m" tutorial script file or the "ex.m" programming example file.
- The coding information provided in the computer-aided laboratory set is sufficient to allow students to switch to programming laboratories at any time.
- Although numbering between the two laboratory sets is consistent, the computer-aided and programming laboratories are different enough to preclude substitution of the tutorial scripts for programming assignments.
- Since laboratories vary quite a bit in length, a summary of comparative requirements is provided in Appendix G. Credit for each laboratory might be assigned on a strict percentage basis.

F. REVISED OBJECTIVES AND REFERENCES

While Stanley [Stanley 82] provides the majority of formulas and techniques, other sources are consulted. Objectives and references for each laboratory are summarized below.

1. Computer-aided Laboratories

- **Computer-aided Laboratory 1 - Signal and Spectrum Generation**

Observe single- and multi-tone signal generation, understanding the role of time and signal vectors [MathWorks 92, p. 164] and the use of plotting commands [MathWorks 92, Chapter 14].

Observe single- and multi-tone one-sided signal spectra, understanding function calls [MathWorks 92, Chapter 16]. Relate spectral characteristics to signal characteristics [Stanley 82, Chapter 2-3].

- **Computer-aided Laboratory 2 - Sampling and Recovery**

Observe the effects of impulse, flat-top, and natural sampling on the spectrum. Calculate sampling periods, bandwidths, and pulse durations. Label spectral components and calculate their amplitudes. Calculate and label the first zero crossings [Stanley 82, Chapters 6-1 and 6-2].

Observe message signal recovery via use of a lowpass filter [Stanley 82, p. 98].

Observe the effects of aliasing on the spectrum and signal recovery [Stanley 82, p. 267].

Observe the effect on the spectrum of varying the duty cycle.

- **Computer-aided Laboratory 3 - Pulse Modulation (PAM, PWM, PPM)**

Observe the differences in the time domain for the three types of modulation. Calculate the sampling period and pulse duration for PAM. Calculate the maximum pulse duration for PWM. Calculate the maximum pulse offset for PPM [Stanley 82, Chapters 6-3 and 6-5].

Observe the differences in the frequency domain for the three types of modulation. Label the sampling frequencies and calculate approximate baseband bandwidths.

- **Computer-aided Laboratory 4 - Analog-to-Digital Conversion and Digital Encoding**

Observe the quantization process and the effect of quantization noise. Calculate dynamic range, actual step size, actual resolution, percentage resolution, and number of levels [Stanley 82, Chapters 7-1 and 7-2].

Observe the process of pulse code modulation (PCM) encoding. Label bit values for non-return-to-zero level (NRZL) unipolar, return-to-zero level (RZL) unipolar, and Manchester coded signals [Stanley 82, Chapter 7-4].

Observe the spectrum of each and calculate approximate baseband bandwidths.

Observe the effect of companding on the quantization process [Stanley 82, Chapter 7-3], via use of a μ -255 compander [Schweber 91].

- **Computer-aided Laboratory 5 - Amplitude Modulation Double Sideband (AM DSB)**

Observe the AM DSB process using a single-tone input. Calculating peak and average power and baseband and transmission bandwidths. Observing coherent detection [Stanley 82, Chapters 4-2, 4-4, 4-7, and 4-8].

Observe the AM DSB process using a multi-tone input. Calculate baseband and transmission bandwidths, and observe coherent detection.

- **Computer-aided Laboratory 6 - Amplitude Modulation Single Sideband (AM SSB)**

Observe the AM SSB process [Couch 93] using a single-tone input. Calculate peak and average power and baseband and transmission bandwidths. Observe coherent detection [Stanley 82, Chapters 4-3, 4-4, 4-7, and 4-8].

Observe the AM SSB process using a multi-tone input. Calculating baseband and transmission bandwidths. Observe coherent detection.

- **Computer-aided Laboratory 7 - Conventional Amplitude Modulation (Conventional AM)**

Observe the conventional AM process using a single-tone input. Calculate peak and average power and baseband and transmission bandwidths [Stanley 82, Chapters 4-5 through 4-8]. Observe envelope detection [Brown 92] [MathWorks 92, p. 310] .

Observe the effect of overmodulating the conventional AM signal.

Observe the conventional AM process using a multi-tone input. Calculate baseband and transmission bandwidths. Observe envelope detection.

- **Computer-aided Laboratory 8 - Frequency Modulation (FM)**

Observe the FM modulation process [Haykin 83] for single-tone input. Calculate peak and average power and baseband and transmission bandwidths (using Carson's rule). Observe control of the FM bandwidth by varying β [Stanley 82, Chapters 5-1 through 5-4]. Calculate and label Δf . Determine distance between the sidebands in the spectrum.

Observe the FM modulation process for multi-tone input. Calculate peak and average power and baseband and transmission bandwidths (using Carson's rule). Observe control of the FM bandwidth by varying Δf .

- **Computer-aided Laboratory 9 - Radio Frequency Digital Modulation Methods (ASK, FSK, BPSK, QPSK)**

Observe the process of amplitude shift keying (ASK) and coherent detection. Calculate bit duration and baseband bandwidth. Label bit values and carrier frequency [Stanley 82, Chapter 7-6].

Observe the process of frequency shift keying (FSK) and coherent detection. Calculate bit duration and baseband bandwidth. Label bit values and carrier frequency [Stanley 82, Chapter 7-6].

Observe the process of binary phase shift keying (BPSK) and coherent detection. Calculate bit duration and baseband bandwidth. Label bit values and carrier frequency [Stanley 82, Chapter 7-6].

Observe the process of quadriphase shift keying (QPSK) and coherent detection. Calculate bit duration and baseband bandwidth. Label bit values and carrier frequency [Stanley 82, Chapter 7-8].

2. Programming Laboratories

- **Programming Laboratory 1 - Signal and Spectrum Generation**

Produce and plot single- and multi-tone signals, controlling signal plots [MathWorks 92, p. 164 and Chapter 14].

Produce and plot spectra, using function calls [MathWorks 92, Ch. 16]. Relate spectral characteristics to signal characteristics [Stanley 82, Chapter 2-3].

- **Programming Laboratory 2A - Natural Sampling and Recovery**

Generate a naturally-sampled signal and its spectrum. Calculate the sampling period, bandwidth, and pulse duration. Label spectral components and calculate their amplitudes [Stanley 82, Chapter 6-1].

Recover the message signal via use of a lowpass filter [Stanley 82, p. 98].

Generate an undersampled signal to observe the effects of aliasing on the spectrum and signal recovery [Stanley 82, p. 267].

Vary the duty cycle of the sampled signal to observe the effect on the spectrum. Calculate and label the first zero crossing.

- **Programming Laboratory 2B - Flatop Sampling and Recovery**

Generate a flatop-sampled signal and its spectrum. Calculate the sampling period, bandwidth, and pulse duration. Label spectral components and calculate their amplitudes [Stanley 82, Chapter 6-1].

Recover the message signal via use of a lowpass filter [Stanley 82, p. 98].

Generate an undersampled signal to observe the effects of aliasing on the spectrum and signal recovery [Stanley 82, p. 267].

Vary the duty cycle of the sampled signal to observe the effect on the spectrum. Calculate and label the first zero crossing.

- **Programming Laboratory 2C - Impulse Sampling and Recovery**

Generate an impulse-sampled signal and its spectrum. Calculate the sampling period and bandwidth. Label spectral components [Stanley 82, Chapter 6-2].

Recover the message signal via use of a lowpass filter [Stanley 82, p. 98].

Generate an undersampled signal to observe the effects of aliasing on the spectrum and signal recovery [Stanley 82, p. 267].

- **Programming Laboratory 3A - Pulse Modulation (PAM and PWM)**

Generate PAM and PWM signals to observe the differences in the time domain for the two types of modulation. Calculate the sampling period and pulse duration for PAM, and maximum pulse duration for PWM [Stanley 82, Chapters 6-3 and 6-5].

Generate the spectra of the PAM and PWM signals to observe the differences in the frequency domain for the two types of modulation. Label sampling frequencies and spectral components. Calculate baseband bandwidths.

- **Programming Laboratory 3B - Pulse Modulation (PAM and PPM)**

Generate PAM and PPM signals to observe the differences in the time domain for the two types of modulation. Calculate the sampling period and pulse duration for PAM, and maximum pulse offset for PWM [Stanley 82, Chapters 6-3 and 6-5].

Generate the spectra of the PAM and PPM signals to observe the differences in the frequency domain for the two types of modulation. Label sampling frequencies and spectral components. Calculate baseband bandwidths.

- **Programming Laboratory 4A - Analog-to-Digital Conversion (Quantization)**

Produce and evaluate the characteristic for two analog-to-digital converters. Calculate dynamic range, actual step size, actual resolution, percentage resolution, and number of levels [Stanley 82, Chapters 7-1 and 7-2].

Sample and quantize a signal using each of the converters.

Measure the quantization noise and compare the noise in the two systems.

- **Programming Laboratory 4B - Pulse Code Modulation (PCM)**

Quantize a signal and generate a bitstream for (PCM) encoding.

Generate the digital signal and spectrum for NRZL unipolar, RZL unipolar, and Manchester coded signals [Stanley 82, Chapter 7-4]. Predict and calculate approximate baseband bandwidths.

- **Programming Laboratory 4C - Companding**

Generate a specified message signal and compress it using a μ -255 compander [Schweber 91]. Understand the purpose of companding [Stanley 82, Chapter 7-3].

Vary the sampling rate and value of μ to reduce quantization noise in a 2-bit unipolar A/D system.

- **Programming Laboratory 5 - Amplitude Modulation Double Sideband (AM DSB)**

Generate single- and multi-tone signals and spectra. Calculate peak and average power and baseband bandwidth for the single-tone signal.

Generate AM DSB signals and spectra. Calculate peak and average power for the single-tone AM DSB signal [Stanley 82, Chapters 4-2, 4-7, and 4-8]. Calculate transmission bandwidth for both signals.

Coherently detect the AM DSB signals [Stanley 82, Chapter 4-4].

- **Programming Laboratory 6 - Amplitude Modulation Single Sideband (AM SSB)**

Generate single- and multi-tone signals and spectra. Calculate peak and average power and baseband bandwidth for the single-tone signal.

Generate AM SSB signals and spectra [Couch 93]. Calculate peak and average power for the single-tone AM SSB signals [Stanley 82, Chapters 4-3, 4-7, and 4-8]. Calculate transmission bandwidth for both signals.

Coherently detect the AM SSB signals [Stanley 82, Chapter 4-4].

- **Programming Laboratory 7 - Conventional Amplitude Modulation (Conventional AM)**

Generate single- and multi-tone signals and spectra. Calculate peak and average power and baseband bandwidth for the single-tone signal.

Generate conventional AM signals and spectra. Calculate peak and average power for the single-tone conventional AM signal [Stanley 82, Chapters 4-5 through 4-8]. Calculate transmission bandwidth for both signals. Perform an envelope detection on each signal [Brown 93] [MathWorks 92, p. 310].

Overmodulate the single-tone conventional AM signal and observe the result.

- **Programming Laboratory 8 - Frequency Modulation (FM)**

Generate single- and multi-tone signals and spectra. Calculate peak and average power and baseband bandwidth for the single-tone signal.

Frequency-modulate the signals [Haykin 83], using $\beta = 10$. Calculate peak and average power and transmission bandwidth (using Carson's rule) for the single-tone signal [Stanley 82, Chapters 5-1 through 5-4]. Calculate and label Δf . Determine distance between the sidebands in the spectrum.

Control bandwidth of the FM signal by varying β , then by varying Δf .

- **Programming Laboratory 9 - Radio Frequency Digital Modulation Methods (ASK, FSK, BPSK, QPSK)**

Generate the NRZL unipolar digital message signal and spectrum, using a random bitstream. Label bit values in the signal. Calculate baseband bandwidth. Generate the amplitude shift keyed (ASK) signal and spectrum. Label bit values and carrier frequency [Stanley 82, Chapter 7-6].

Generate the frequency shift keyed (FSK) signal and spectrum. Label bit values and carrier frequencies [Stanley 82, Chapter 7-6].

Generate the NRZL bipolar digital message signal and spectrum, using a random bitstream. Label bit values in the signal. Calculate baseband bandwidth. Generate the binary phase shift keyed (BPSK) signal and spectrum. Label bit values and carrier frequency [Stanley 82, Chapter 7-6].

Split the signal into odd and even parallel signals. Generate the quadriphase shift keyed (QPSK) signal and spectrum. Label bit values and carrier frequency [Stanley 82, Chapter 7-8].

V. CONCLUSION

The success of this laboratory development effort cannot currently be measured due to the fact that only about 70% of the initial laboratories have been evaluated in a classroom setting. This figure translates to only 40% of the final laboratory sets.

But in other ways, the development effort can be considered successful. It is a walking advertisement for rapid prototyping—the difficulties inherent with “build as you go” resulted in a product much closer to the users’ needs than they themselves initially envisioned. Given the adaptive nature of rapid prototyping, the process was no more chaotic than could be expected. It must be noted, however, that the rapid prototyping applied essentially to the laboratory contents and not to the Communications Toolbox functions. It would have been advantageous if development of the Communications Toolbox had been completed prior to the quarter in which the laboratories were developed.

So while the development effort can be considered successful, an evaluation of the total success of this project is premature. In fact, it may be ripe for judgement only after graduates have completed follow-on tours. For the immediate future, however, the developer is confident that the product is responsive to the needs of the users.

REFERENCES

- [Brown 93] Department of Electrical and Computer Engineering, Naval Postgraduate School Technical Report no. NPSEC-93-017, *SPC Toolbox*, by LT Dennis W. Brown and Monique P. Fargues, p. 55, 15 October 1993.
- [Couch 93] Couch, Leon W., II, *Digital and Analog Communication Systems*, Macmillan Publishing Company, 1993, p. 314.
- [Haykin 83] Haykin, Simon, *Communication Systems*, Second Edition, John Wiley & Sons Inc., 1983, p. 190.
- [MathWorks 92] The MathWorks, Inc., *The Student Edition of MATLAB for Macintosh Computers*, Prentice-Hall, Inc., 1992, Chapters 14, 16, and 19, and Part 4.
- [Moose 93] Moose, Paul (Chairman, C3 Academic Group, Naval Postgraduate School) memorandum, Subject: Final Report of the C3 Futures Committee, 3 Feb 93.
- [Schweber 91] Schweber, William, *Electronic Communications Systems, A Complete Course*, Prentice-Hall, Inc., 1991, p. 342.
- [Stanley 82] Stanley, William D., *Electronic Communications Systems*, Prentice-Hall, Inc., 1982, Chapters 2, 3, 4, 5, 6, and 7.

**This page is intentionally
left blank.**

APPENDIX A—COMPUTER-AIDED LABORATORIES

Name: _____

Section: _____

EO 3513 Computer-aided Laboratory 1 *Signal and Spectrum Generation*

This laboratory introduces the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

While the laboratories in this set require you only to run MATLAB script files, you will want to be familiar with some basic plotting and printing commands, and understand the mechanics of script writing and function calls.

The m-file for this laboratory is "lab1scr.m.". It may be helpful to print the m-file for reference before running the script.

Part 1—Observe signal generation

A. Establishing the time vector

Since signals are functions of time, a time vector must be established prior to generating a signal. Below is the time vector "t1" that is one second in duration and has a "step size" of one-thousandth of a second. The variable name "delta_t" (Δt) is usually assigned to the step size throughout this series of laboratories. The time vector consists of 1,001 values, or points, starting at 0 and ending at 1.

```
t1=0:0.001:1; %time vector
```

B. Generating a signal

The single-tone periodic signal s1, generated using the formula below, has a frequency of 10 Hz:

```
s1=cos(2*pi*10*t1); %single-tone signal
```

A multi-tone signal can be generated by adding sinusoids together. The signal s2 contains frequencies of 20,35, and 50 Hz:

```
s2=10*cos(2*pi*20*t1)+4*cos(2*pi*35*t1)+6*cos(2*pi*50*t1); %multi-tone signal
```

In future laboratories you will use the maximum amplitude of a signal in order to calculate its power in the time domain. The signal "s2" has a maximum amplitude of 20 (in this case, conveniently found by adding the maximum amplitudes of its three cosines). If not easily determined from the formula or the signal plot, use the "max" command in MATLAB to find the approximate maximum signal amplitude:

```
max_of_s=max(s); %maximum amplitude in the signal s
```

The "plot", "title", "xlabel", and "ylabel" commands, used as shown below, produce clearly labeled graphs of the signals:

```
plot(t1,s1)
title('Signal')
xlabel('Time')
ylabel('Amplitude')
```

Run the script "lab1scr.m."

Plot 1: Observe the single-tone signal s1 plotted against the time vector t1.

Plot 2: Observe the multi-tone signal s2 plotted against the time vector t1.

C. Controlling signal plots

Signals s1 and s2 are lower-frequency signals than will commonly be used in these laboratories. The vectors t2, s3, and s4 below are representative of the time and signals vectors that you will be observing.

```
t2=0:0.0001:1; %time vector of 10,001 points
s3=5*sin(2*pi*200*t2); %single-tone signal
s4=10*cos(2*pi*130*t2)+5*cos(2*pi*335*t2)+cos(2*pi*400*t2); %multi-tone signal
```

Using higher frequencies makes signal characteristics hard to distinguish when the entire signal vector is plotted. Described below are two methods of limiting plot size to 1000 points.

(1) Restrict the number of points plotted. This method has an advantage in that the two vectors need not be the same length:

```
plot(t2(1:1000),s3(1:1000))
```

(2) Use the "axis" command to freeze the axes at minimum and maximum values of time, amplitude, etc, (as opposed to numbers of points). Following use of the axis freeze, the axis must be released by typing the command "axis".

One signal will frequently be plotted over another in order to compare signals (a recovered signal and its message signal, for example). Three methods are described below.

- (1) Listing pairs of x and y arguments for the "plot" command:

```
plot(t2(1:500),s3(1:500),t2(1:500),s4(1:500)) %plot s4 over s3
```

- (2) Using the "hold on" and "hold off" commands, sometimes combined with "pause":

```
plot(t2,s4) %plot signal with larger amplitude first
hold on
pause(3) %wait 3 seconds
plot(t2,s3,'g') %plot in green
hold off
```

- (3) Creating a signal matrix to plot against a single time vector:

```
plot(t2(1:1000),[s3(1:1000);s4(1:1000)])
```

Other commands that affect plotting of results that you may see in the scripts include the following:

"Subplot" permits up to four plots per graphics window.

"Pause" with no argument delays program execution until the user presses return (preventing graphs from whizzing by unobserved)

"Clg" between sets of plots prevents plots from being superimposed.

Press return to continue.

Plot 3: Observe s4; press return; and observe s3 plotted over it. These signals are restricted to 500 points.

D. Printing plots

On PC platforms, the command "meta" creates a graphics file. "Prtsc" dumps the current graph window to a printer. "Print" sends a high-resolution copy to the printer. (The capability to use these commands may vary according to machine and software configuration.) You may need to edit the script file in order to print a plot.

For Macintosh platforms, choose "print" from the "file" menu to print the active graph window, or use the "save as" command in the "file" menu to create a Quick-Draw graphics file.

Part 2—Observe spectrum generation

A. Calling a function

The Communications Toolbox contains the function “spectral,” with the following function call:

```
[specsig,Hz,fftsig]=spectral(s,delta_t); %function call to spectral.m
```

“Spectral” produces a one-sided spectrum. A two-sided spectrum with relatively-correct amplitudes could also be generated, using the following command:

```
two_sided_spec=fftshift(abs(fftsig)); %plot against the time vector
```

The step size and number of points in the time vector relate directly to (1) the number of frequencies observed in the spectrum, and (2) the resolution of the spectrum. The above vector of 1,001 points with $\Delta t=0.001$ produces a one-sided spectrum of 500 Hz—half the number of points—with a resolution of approximately 1 Hz.

Vector lengths for spectral plots are usually not reduced in order to see as many of the spectral components as possible.

Press return to continue.

Plot 4: Observe the spectrum for s1.

Label the Hz value and the amplitude of the spectral component.

Press return to continue.

Plot 5: Observe the spectrum for s2.

Label the Hz values and the amplitudes of the spectral components.

Plot 6: Observe the spectrum for s4.

Label the Hz values and the amplitudes of the spectral components.

Question 1: Compare Plots 5 and 6. Why does Plot 6 display more frequencies than Plot 5?

Question 2: Given the spectral plot of a multi-tone signal, how could you determine the amplitudes and frequencies of each of the signal tones?

Name: _____

Section: _____

EO 3513 Computer-aided Laboratory 2

Sampling and Recovery

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

The m-file for this laboratory is "lab2scr.m." It may be helpful to print the m-file for reference before running the script.

Part 1—Observe the effects of sampling on the spectrum

A. Generating the signal

You will be observing three types of sampling on the following message signal:

```
t=0:.0001:1;           %time vector
s=cos(2*pi*120*t)+cos(2*pi*300*t)+cos(2*pi*450*t); %multi-tone signal
```

Run the script "lab2scr.m."

Plot 1: **Observe the signal s plotted against the time vector t.**

B. Sampling the signal

Signals are sampled by calling the functions "natsamp.m," "impsamp.m," and "flattop.m." Since 450 Hz is the highest frequency in the message signal, the sampling rate of 1000 Hz adequately prevents aliasing.

The naturally-sampled signal is produced by multiplying a pulse train and the message signal.

Press return to continue.

Plot 1: **Observe the pulse train, at a rate of 1000 Hz and duty cycle of 0.5, plotted over the message signal.**

Plot 2: Observe the naturally-sampled signal, at a sampling rate of 1000 Hz and duty cycle of 0.5.

The impulse-sampled signal is produced by multiplying an impulse train and the message signal.

Press return to continue.

Plot 3: Observe the impulse train, at a sampling rate of 1000 Hz, plotted over the message signal.

Plot 4: Observe the impulse-sampled signal, at a sampling rate of 1000 Hz.

The flattop-sampled signal is produced by convolving the impulse train with a single flattop pulse.

Press return to continue.

Plot 5: Observe the flattop-sampled signal, at a sampling rate of 1000 Hz and duty cycle of 0.5.

Question 1: Calculate the following values, in seconds, for the naturally- and flattop-sampled signals:

sampling period T
pulse duration τ

Question 2: Describe the distinguishing pulse shape of each sampled signal:

naturally-sampled
flattop-sampled
impulse-sampled

C. Generating the spectrum

The one-sided spectrum of each sampled signal is produced using the function "spectral.m."

Press return to continue.

Plot 6: Observe the spectrum of the message signal.

Label the Hz value of each of the baseband signal frequencies.

Plot 7: Observe the spectrum of the naturally-sampled signal.

Label the following groups of frequencies in the spectrum:

baseband signal frequencies
spectral components associated with the sampling
frequency (f_s) and each of its multiples ($2f_s$, $3f_s$, etc.)

Label the first zero crossing ($1/\tau$).

Press return to continue.

Plot 8: Observe the spectrum of the impulse-sampled signal.

Label the following groups of frequencies in the spectrum:

baseband signal frequencies
spectral components associated with the sampling
frequency (f_s) and each of its multiples ($2f_s$, $3f_s$, etc.)

Plot 9: Observe the spectrum of the flat-top-sampled signal.

Label the following groups of frequencies in the spectrum:

baseband signal frequencies
spectral components associated with the sampling
frequency (f_s) and each of its multiples ($2f_s$, $3f_s$, etc.)

Label the first zero crossing ($1/\tau$).

Question 3: Describe the overall shape of each spectrum. Does each of the spectral plots conform to your theoretical expectations? Note any discrepancies.

The amplitude spectrum of a naturally-sampled signal can be determined using the formula

$$X_s(f) = \sum_{N=0}^{\infty} P_N X(f - Nf_s)$$

where

$$P_N = \frac{d \sin(N\pi d)}{N\pi d}$$

and d is the duty cycle and N indicates the number of the harmonic.

Question 4: Calculate P_N for $N = 1$, $N = 2$, and $N = 3$.

Compare with the values shown on the spectral plot for natural sampling.

The amplitude spectrum of a flattop-sampled signal can be determined using the formula

$$X_s(f) = \sum_{N=0}^{\infty} P_N X(f - Nf_s)$$

where

$$P_N = \frac{d \sin(N\tau f)}{N\tau f}$$

and d is the duty cycle, τ is the pulse duration in seconds, and f indicates the frequency in Hz.

Question 5: Calculate P_N for $f = 550$, $f = 700$, and $f = 880$.

Compare with the values shown on the spectral plot for flattop sampling.

Part 2—Observe the message signal recovery

Signals are recovered using the function "recovers.m."

A. Recovering the message signal

An ideal lowpass filter at 500 Hz retains the baseband frequencies while filtering out the frequencies produced by sampling. Recovery of the message signal will be shown using the naturally-sampled signal. In this "perfect" system, you will observe a nearly perfect recovery.

Press return to continue.

Plot 10: Observe the recovered signal plotted over the message signal.

Part 3—Observe the effects of aliasing on the spectrum and recovery

A. Generating undersampled signals

An undersampled signal is sampled less than twice the highest signal frequency, producing an effect known as "aliasing." Undersampling will be demonstrated using natural sampling at a rate of 600 Hz.

Question 6: What is the minimum theoretical sampling frequency for the message signal s ?

Press return to continue.

Plot 11: Observe the undersampled (naturally-sampled) signal.

Plot 12: Observe the undersampled (naturally-sampled) signal spectrum.

Press return to continue.

Plot 13: Observe the recovered undersampled (naturally-sampled) signal plotted over the message signal.

Label the recovered signal.

Question 7: Compare the undersampled signal spectrum in plot 12 to its counterpart in plot 7. What is the effect of undersampling on the spectrum? What is the effect of undersampling on the signal recovery?

Part 4—Observe the effect on the spectrum of varying the duty cycle

A. Generating the sampled signal

The flat-top-sampled signal is used to demonstrate the effects of varying the duty cycle.

Press return to continue.

Plot 14: Observe the flat-top-sampled signal, sampled at 1000 Hz with a duty cycle of 0.7.

Plot 15: Observe the flattop-sampled signal spectrum, sampled at 1000 Hz with a duty cycle of 0.7.

Label the Hz value at the first zero crossing ($1/\tau$).

Press return to continue.

Plot 16: Observe the flattop-sampled signal, sampled at 1000 Hz with a duty cycle of 0.3.

Plot 17: Observe the flattop-sampled signal spectrum, sampled at 1000 Hz with a duty cycle of 0.3.

Label the Hz value at the first zero crossing ($1/\tau$).

Question 8: What is the effect of reducing the duty cycle on the sampled signal baseband bandwidth?

Name: _____

Section: _____

EO 3513 Computer-aided Laboratory 3

Pulse Modulation (PAM, PWM, PPM)

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

The m-file for this laboratory is "lab3scr.m." It may be helpful to print the m-file for reference before running the script.

Part 1—Observe the differences in the time domain for the three types of modulation

A. Generating the signal

You will be observing the three types of pulse modulation on the following message signal:

```
t=0:.0001:1; %time vector to one second
s=5*(cos(2*pi*75*t)+sin(2*pi*150*t)); %signal
```

Run the script "lab3scr.m."

Plot 1: Observe the signal *s* plotted against the time vector *t*.

B. Modulating the signal

Signals will be modulated using the functions "flattop.m," "pulswid.m," and "pulspos.m." To allow close examination of pulse widths and positions, a sampling rate of 500 Hz will be used.

You studied the flattop-sampled signal in Laboratory 2, and saw that while its pulses varied in amplitude, they always appeared at the start of the sampling period *T*, and had a constant duration τ . Flattop sampling is one implementation of pulse-amplitude modulation (natural sampling is the other). In this laboratory, the familiar characteristics of the pulse-amplitude modulated (PAM) signal will be compared to characteristics of the pulse-width and pulse-position modulated signals.

Press return to continue.

Plot 1: Observe the PAM signal ($d = 0.5$) plotted over the message signal.

Question 1: Calculate the following values, in seconds, for the PAM signal:

sampling period T
pulse duration τ

Like PAM signals, pulse-width modulated (PWM) signal pulses begin with the sampling period T . Pulse amplitudes are constant, while their widths vary, based on the amplitude of the message signal at each pulse beginning. To facilitate this variation in pulse width, the maximum pulse duration is expressed as a fraction of the sampling period T . The signal you will observe has a maximum pulse duration of 0.8, meaning that the widest pulse that could occur would be 0.8 of the sampling period T in duration, where maximum signal amplitude fell at the beginning of a pulse. The most narrow pulse would occur where the minimum signal amplitude fell at the beginning of a pulse.

Question 2: The maximum signal amplitude of s is 8.8, the modulation rate is 500 Hz, and the maximum pulse duration is 0.8 of the sampling period. Calculate the duration in seconds of the widest pulse that could occur for its PWM signal.

Question 3: The zero crossings in this signal occur halfway between the minimum and maximum signal values. Calculate the pulse duration in seconds for a PWM pulse that occurs at the beginning of a zero crossing.

Press return to continue.

Plot 2: Observe the PWM (maximum pulse duration = 0.8) signal plotted over the message signal.

The maximum signal amplitude in s occurs at 0.028 seconds; a zero crossing occurs at 0.03 seconds. Verify your answers to Questions 2 and 3.

Label the duration of each of these pulses on the plot.

A pulse-position modulated signal, like a pulse-amplitude modulated signal, has a constant duration τ . However, the pulse beginnings vary in location within the sampling period T . Most PPM systems vary pulse position from the middle of the sampling period T . Negative signal amplitudes cause the pulse to shift left; positive signal amplitudes cause the pulse to shift right.

The function "pulspos.m" transmits signal information via the amount of the pulse offset from the beginning of the sampling period. To allow for larger variations in the pulse offset, the pulse duration τ is kept small. The PPM signal you will observe has a duty cycle of 0.1; thus the largest pulse offset that could occur would be 0.9 of the sampling period T , observed at the maximum signal amplitude. No pulse offset occurs at the minimum signal amplitude.

In the following plot, the sampling period T is defined by the grid imposed over the plot.

Press return to continue.

Plot 3: Observe the pulse-position modulated signal ($d = 0.1$) plotted over the message signal. Note the pulse positions at the signal maximum (0.028 seconds), signal minimum (0.032 seconds), and zero crossing (0.03 seconds) values.

Part 2—Observe the differences in the frequency domain for the three types of modulation

A. Generating the spectra

The one-sided spectrum of each modulated signal is produced using the function "spectral.m."

Press return to continue.

Plot 4: Observe the spectrum of the message signal.
Label the Hz values of the baseband signal frequencies.

Plot 5: Observe the spectrum of the PAM signal.
Label the Hz value of the sampling frequency (fs).

Press return to continue.

Plot 6: Observe the spectrum of the PWM signal.
Label the Hz value of the sampling frequency (fs).

Plot 7: Observe the spectrum of the PPM signal.
Label the Hz value of the sampling frequency (fs).

B. Calculating baseband bandwidth

Recall that while PAM signals have a baseband bandwidth of approximately $0.5/\tau$, PPM and PWM signals have larger baseband bandwidths, approximately $0.5/\text{risetime}$. Consider the risetime of the pulses in your modulated signals to be equal to the step size in the time vector.

Question 4: Using the above approximations, calculate the baseband bandwidths for the PAM, PWM and PPM signals. Do these values reflect what you observe in the spectral plots? Note any discrepancies.

Name: _____

Section: _____

EO 3513 Computer-aided Laboratory 4

Analog-to-Digital Conversion and Digital Encoding

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

The m-file for this laboratory is "lab4scr.m." It may be helpful to print the m-file for reference before running the script.

Part 1—Observe the quantization process and the effect of quantization noise

A. Generating the signal

You will be observing the quantization process on the following message signal:

```
t=0:.0001:1;           %time vector to one second
s=5*cos(2*pi*50*t)+4*cos(2*pi*90*t); %signal
```

The function "quantize.m" is used both to set the characteristic for a bipolar quantization system, and to quantize input signals. You will first be observing a 5-bit bipolar offset converter that will quantize signals at values between -10 and +10 volts.

B. Evaluate the converter

Run the script "lab4scr.m."

Plot 1: Observe the quantization characteristic for the converter.

Question 1: Calculate the following values relating to the quantization characteristic for this system:

dynamic range
actual step size
actual resolution
percentage resolution
number of levels

Would you describe this quantizer as “mid-step,” or “mid-tread”?

C. Compare the sampled and quantized signals

Signals are typically sampled prior to quantizing. “Flattop.m” is used to sample the signal at a rate of 200 Hz with a duty cycle of 0.5.

Press return to continue.

Plot 2: Observe the sampled signal plotted over the message signal.

The signal is then quantized at the same rate, 200 Hz. The quantized signal is assigned a bin number for each sample. Bin numbers for this quantized signal range from 0 to 31.

Plot 3: Observe the quantized signal plotted over the sampled signal.

From the command window, obtain the voltage value for each level of the converter (“quanch_y”), and the bin numbers for the first 6 samples in the plot (“bin_nums”).

Question 2: List the amplitude (“voltage”) of the quantized signal in each of the first 6 sampling periods.

D. Measuring the quantization noise

The difference between the message signal and the quantized signal is referred to as “quantization noise”—noise introduced by the quantization process. The function “snr.m” returns the signal to noise ratio, measured in dB.

Press return to continue.

Plot 4: Observe the quantized signal plotted over the message signal.

Question 3: From Plot 4, obtain the value of the signal-to-noise ratio for the quantized signal. Record this value.

Part 2—Observe the process of PCM (pulse code modulation) encoding

A. Generating a binary-encoded signal

The function "encode.m" is used to convert "bin_nums," the vector returned from "quantize.m" containing the quantization levels, to "codedsig," a bitstream of 1's and 0's.

The vector "codedsig" is passed to the functions "nrzluni.m," "rzuni.m," and "manchest.m," which translate the bitstream into PCM signals.

In order for these PCM-encoded signals to transmit information at the same rate as the sampled signal, the bit rate of the PCM signals must be five times as high as the sampling rate (1000 vice 200) in order to efficiently transmit the five bits (elements) used in the encoding scheme.

One-sided spectral plots are generated using the function "spectral.m."

B. Generating a non-return-to-zero level (NRZL) unipolar coded signal

Press return to continue.

Plot 5: Observe the NRZL unipolar coded signal plotted over the quantized signal.

Plot 6: Observe an expanded view of the first two words of the NRZL unipolar coded signal.

Label the bit values (0 or 1) in these two words. (Refer to "bin_nums" for the base 10 values.)

Press return to continue.

Plot 7: Observe the NRZL unipolar coded signal spectrum. Note the DC value present in the spectrum.

C. Generating a return-to-zero level (RZL) unipolar coded signal

Press return to continue.

Plot 8: Observe the RZL unipolar coded signal plotted over the quantized signal.

Plot 9: Observe an expanded view of the third and fourth words of the RZL unipolar coded signal.

Label the bit values (0 or 1) in these two words.

Press return to continue.

Plot 10: Observe the RZL unipolar coded signal spectrum.

D. Generating a manchester coded signal

Press return to continue.

Plot 11: Observe the manchester coded signal plotted over the quantized signal.

Plot 12: Observe an expanded view of the fifth and sixth words of the manchester coded signal.

Label the bit values (0 or 1) in these two words.

Press return to continue.

Plot 13: Observe the manchester coded signal spectrum. Note the absence of DC value present in the spectrum.

Question 4: From the command window, obtain the first 30 values of "codedsig." Record these values. Is this bit pattern reflected on Plots 6, 9, and 12?

Press return to continue.

Question 5: Describe the distinguishing characteristics of each encoding scheme:

**NRZL unipolar
RZL unipolar
manchester**

E. Estimate bandwidth for the PCM signals

The minimum theoretical PCM bandwidth for sinc-shaped pulses is $B \cdot N$, the baseband message signal bandwidth times the number of elements(bits). Rectangular pulses theoretically require an *infinite* bandwidth, but can be estimated based on τ , the pulse duration:

$$B = 0.5/\tau$$

The value of τ depends on the PCM encoding scheme employed. For NRZL coded signals, τ is equal to the bit duration. For RZL and manchester coded signals, τ is equal to $1/2$ of the bit duration.

Question 6: Calculate the approximate baseband bandwidth for the PCM signals:

**NRZL unipolar coded signal
RZL and manchester coded signals**

Do these values reflect what you observe in the spectral plots?

Part 3—Observe the effects of companding on the quantization process

A. Generating a signal

The effects of the companding process will be observed upon the following signal:

```
=0:.0001:0.1; %time vector to 0.1 seconds  
s=2+2.1*cos(2*pi*50*t)+1.7*cos(4*pi*50*t)+1.5*cos(6*pi*50*t); %signal for  
companding  
s=s+1.3*cos(8*pi*50*t);
```

Press return to continue.

Plot 14: Observe the plot of the message signal. Notice the low-level signal activity.

B. Sampling and quantizing the signal (without companding)

The signal is sampled using "flattop.m" at a rate of 1500 Hz with a duty cycle of 0.5.

Press return to continue.

Plot 14: Observe the sampled signal plotted over the message signal.

This sampled signal is quantized using the function "quantuni.m." This unipolar quantizing function accepts signals between 0 and 10 volts, and uses truncation rather than rounding. The characteristic of a 3-bit binary converter is achieved by passing in two symbols and three elements, resulting in 8 levels. The function "snr.m" is used to calculate the signal to noise ratio.

Plot 15: Observe the quantized signal plotted over the sampled signal. Notice the changes in the signal level that are not captured by the converter.

Press return to continue.

Plot 16: Observe the message signal plotted over the quantized signal.

Question 7: From Plot 16, obtain the value of the signal-to-noise ratio for the quantized signal. Record this value.

C. Compare compression characteristics for values of μ

Companding (the process of compressing, then expanding) improves the quantization process by proportioning signals that spend most of the time in the lower range of the dynamic range. The functions "compress.m" and "expand.m" simulate a μ -255 compander. Values of μ range from 1 to 255.

"Compress.m" will be used to compress the signal two separate times.

Press return to continue.

Plot 17: Observe the plot of the message signal, compressed with a value of $\mu = 255$.

Plot 18: Observe the plot of the message signal, compressed with a value of $\mu = 10$.

Question 8: What is the effect of compression on the signal?

D. Sampling and quantizing the compressed signal

The compressed signal ($\mu = 10$) is sampled at a rate of 1500 Hz with a duty cycle of 0.5.

Press return to continue.

Plot 19: Observe the sampled compressed signal plotted over the compressed message signal.

The sampled compressed signal is quantized at a rate of 1500 Hz, again using the characteristic of a 3-bit converter.

Plot 20: Observe the quantized compressed signal plotted over the sampled compressed signal.

E. Expanding the quantized compressed signal

The function "expand.m" is used to expand the quantized compressed signal.

Press return to continue.

Plot 21: Observe the companded signal plotted over the message signal.

Question 9: From Plot 21, obtain the value of the signal-to-noise ratio for the companded signal. Record this value and compare it to the ratio obtained in Question 7. Did the companding process reduce the amount of quantization noise?

**This page is intentionally
left blank.**

Name: _____

Section: _____

EO 3513 Computer-aided Laboratory 5

Amplitude Modulation Double Sideband (AM DSB)

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

The m-file for this laboratory is "lab5scr.m." It may be helpful to print the m-file for reference before running the script.

Part 1—Observe the double sideband amplitude modulation (AM DSB) process using a single-tone input

A. Generating the signal and spectrum

You will be observing the AM DSB process on the following single-tone message signal:

```
t=0:.0001:1;           %time vector to one second
s=15*cos(2*pi*150*t);   %single-tone signal
```

Parseval's theorem states that average signal power can be calculated in either the time or the frequency domain. The following formula applies to calculation in the time domain:

$$P = A_0^2 + \frac{1}{2} \sum_{N=1}^{\infty} (A_N^2 + B_N^2)$$

Peak power is calculated as follows:

$$P_P = \frac{A_P^2}{2}$$

Question 1: Calculate the following values for the single-tone message signal:

**peak power
average power
baseband bandwidth**

Run the script "lab5scr.m."

Plot 1: Observe the message signal.

"Spectral.m" is used to generate spectra.

Plot 2: Observe the spectrum of the message signal.

Label the following values:

**Hz value of the spectral component
amplitude of the spectral component
baseband signal bandwidth in Hz**

B. Observe the single-tone AM DSB signal

The message signal is modulated by multiplying it with a cosine with a carrier frequency of 2000 Hz.

Peak power for the AM DSB signal is calculated as before:

$$P_P = \frac{A_P^2}{2}$$

Average power for the AM DSB signal is obtained using by adding the power produced by each of the two components, resulting in

$$P = \frac{A^2}{4}$$

Question 2: Predict the following values for the single-tone AM DSB signal:

**peak power
average power
transmission bandwidth**

Press return to continue.

Plot 3: Observe the plot of the AM DSB signal.

Computer-aided Laboratory 5—page 2

Plot 4: Observe an expanded view of the AM DSB signal.

Label the phase shifts shown in this portion of the signal.

C. Verify the power and bandwidth of the AM DSB signal

Average power was calculated in the time domain; it will be verified in the frequency domain. The function "psd.m" is used to generate the power spectral density of the modulated signal. The power levels associated with each frequency are added using the "sum" command to produce average signal power.

Press return to continue.

Plot 5: Observe the AM DSB signal spectrum.

Label the following values:

**amplitude of each spectral component
Hz value of each spectral component
transmission bandwidth in Hz**

Plot 6: Observe the AM DSB power spectral density.

Question 3: From Plot 6, obtain the values representing peak and average power for the signal-tone signal, and record them.

Do your calculations for bandwidth and power agree with the computer-generated values and spectrum?

D. Observe the recovery of the AM DSB signal

AM DSB signals are recovered using a three-step process:

- 1) Modulate the DSB signal by the carrier (called "demodulation")
- 2) Use a lowpass filter to recover the signal frequencies in the baseband
- 3) Multiply by a factor of 2 to restore signal amplitude

The function "recoverm.m" is used to lowpass-filter the baseband frequencies with a cutoff frequency of 160 Hz. In this "perfect system," you will observe an almost perfect recovery.

Question 4: Why is coherent detection (detection using the carrier) necessary for an AM DSB signal?

Press return to continue.

Plot 7: Observe the demodulated AM DSB signal (prior to filtering).

Plot 8: Observe the AM DSB recovered signal plotted over the message signal.

Press return to continue.

Plot 9: Observe the recovered amplified AM DSB signal plotted over the message signal.

Press return to continue.

“Cmplxenv.m” is used to perform an envelope-detection on the AM DSB signal, demonstrating the result of using an inappropriate detection method.

Plot 9: Observe the expanded view of the AM DSB signal and message signal.

Press return to view the envelope-detected signal.

Part 2—Observe the double sideband amplitude modulation (AM DSB) process using a multi-tone input

A. Generating the signal and spectrum

Next you will be observing the AM DSB process on the following multi-tone message signal:

$s=10*\cos(2*\pi*350*t)+12*\cos(2*\pi*220*t)+20*\cos(2*\pi*100*t);$ %multi-tone signal

Press return to continue.

Plot 10: Observe the multi-tone message signal.

Plot 11: Observe the spectrum of the message signal.

Label the following values for the message signal:

Hz value of each spectral component
amplitude of each spectral component
baseband bandwidth in Hz

B. Observe the multi-tone AM DSB signal and spectrum

The message signal is again modulated by multiplying it with a cosine with a carrier frequency of 2000 Hz.

Press return to continue.

Plot 12: Observe the plot of the AM DSB signal.

Plot 13: Observe an expanded view of the AM DSB signal.

Label the phase shifts shown in this portion of the signal

Press return to continue.

Plot 14: Observe the AM DSB signal spectrum.

Label the following values:

**Hz value of each spectral component
amplitude of each spectral component
transmission bandwidth in Hz**

C. Observe the recovery of the AM DSB signal

The three-step recovery process is repeated to recover the multi-tone AM DSB signal. The ideal lowpass filter has a cutoff frequency of 360 Hz.

Press return to continue.

Plot 15: Observe the demodulated AM DSB signal (prior to filtering).

Plot 16: Observe the AM DSB recovered signal plotted over the message signal.

Press return to continue.

Plot 16: Observe the recovered amplified AM DSB signal plotted over the message signal.

**This page is intentionally
left blank.**

Name: _____

Section: _____

EO 3513 Computer-aided Laboratory 6

Amplitude Modulation Single Sideband (AM SSB)

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

The m-file for this laboratory is "lab6scr.m." It may be helpful to print the m-file for reference before running the script.

Part 1—Observe the single sideband amplitude modulation (AM SSB) process using a single-tone input

A. Generating the signal and spectrum

You will be observing the AM SSB process on the following single-tone message signal:

```
t=0:.0001:1;           %time vector to one second
s=15*cos(2*pi*130*t);   %single-tone signal
```

Parseval's theorem states that average signal power can be calculated in either the time or the frequency domain. The following formula applies to calculation in the time domain:

$$P = A_0^2 + \frac{1}{2} \sum_{N=1}^{\infty} (A_N^2 + B_N^2)$$

Peak power is calculated as follows:

$$P_P = \frac{A_P^2}{2}$$

Question 1: Calculate the following values for the single-tone message signal:

**peak power
average power
baseband bandwidth**

Run the script "lab6scr.m."

Plot 1: Observe the message signal.

"Spectral.m" is used to generate spectra.

Plot 2: Observe the spectrum of the message signal.

Label the following values for the message signal:

**Hz value of the spectral component
amplitude of the spectral component
baseband signal bandwidth in Hz**

B. Observe the single-tone AM SSB signals and spectra

Single sideband modulation could be accomplished, in theory, by double sideband modulation followed by filtering of unwanted frequencies. In practice, however, retaining one sideband while rejecting the other is a complex procedure. In this laboratory, a Hilbert transform, simulated by the function "hilbert.m," will be used to apply a 90° phase shift to the signal, cancelling either the upper or lower sideband.

"Hilbert.m" is called by the function "ssb.m," which is used to generate the lower and upper sideband signals at a carrier frequency of 3000 Hz.

Peak power for the AM SSB signal is calculated as before:

$$P_p = \frac{A_p^2}{2}$$

Average power for the AM SSB signal is calculated as follows:

$$P = \frac{A^2}{2}$$

An examination of the AM SSB signal plots will confirm that the signal maximum in an AM SSB signal is half of the signal maximum in its message signal.

Question 2: Predict the following values for the single-tone AM SSB signal:

**peak power
average power
bandwidth**

Press return to continue.

Plot 3: Observe the plot of the AM lower sideband (LSB) signal.

Plot 4: Observe an expanded view of the AM LSB and message signals.

Press return to continue.

Plot 5: Observe the plot of the AM upper sideband (USB) signal.

Plot 6: Observe an expanded view of the AM USB and message signals.

C. Verify the power and bandwidth of the AM SSB signals

Average power was calculated in the time domain; it will be verified in the frequency domain. The function "psd.m" is used to generate the power spectral density of the modulated signals. The power levels associated with each frequency are added using the "sum" command to produce average signal power.

Press return to continue.

Plot 7: Observe the AM LSB signal spectrum.

Label the following values:

**amplitude of each spectral component
Hz value of each spectral component
bandwidth in Hz**

Plot 8: Observe the AM USB signal spectrum.

Label the following values:

**amplitude of each spectral component
Hz value of each spectral component
bandwidth in Hz**

Press return to continue.

Plot 9: Observe the AM LSB power spectral density.

Plot 10: Observe the AM USB power spectral density.

Question 3: From Plots 9 and 10, obtain the values representing peak and average power for the signal-tone signal, and record them.

Do your calculations for bandwidth and power agree with the computer-generated values and spectrum?

D. Observe the recovery of the AM SSB signals

AM SSB signals are recovered using a three-step process:

- 1) Modulate the SSB signal by the carrier (called "demodulation")
- 2) Use a lowpass filter to recover the signal frequencies in the baseband
- 3) Multiply by a factor of 4 to restore signal amplitude

The function "recoverm.m" is used to lowpass-filter the baseband frequencies with a cutoff frequency of 150 Hz. In this "perfect system," you will observe an almost perfect recovery.

Press return to continue.

Plot 11: Observe the demodulated AM LSB signal (prior to filtering).

Plot 12: Observe the demodulated AM USB signal (prior to filtering).

Press return to continue.

Plot 13: Observe the AM LSB recovered signal plotted over the message signal.

Press return to continue.

Plot 13: Observe the recovered amplified AM LSB signal plotted over the message signal.

Plot 14: Observe the AM USB recovered signal plotted over the message signal.

Press return to continue.

Plot 14: Observe the recovered amplified AM USB signal plotted over the message signal.

Part 2—Observe the AM SSB process using a multi-tone input

A. Generating the signal and spectrum

Next you will be observing the AM SSB process on the following multi-tone message signal:

$s=5*\cos(2*\pi*400*t)+12*\cos(2*\pi*230*t)+20*\cos(2*\pi*170*t);$ %multi-tone signal

Press return to continue.

Plot 15: Observe the message signal.

Plot 16: Observe the spectrum of the message signal.

Label the following values for the message signal:

**Hz value of each spectral component
amplitude of each spectral component
baseband bandwidth in Hz**

B. Observe the multi-tone AM SSB signals and spectra

The message signal is modulated by the function "ssb.m" at carrier frequency of 3000 Hz.

Press return to continue.

Plot 17: Observe the plot of the AM LSB signal.

Plot 18: Observe an expanded view of the AM LSB and message signals.

Press return to continue.

Plot 19: Observe the plot of the AM USB signal.

Plot 20: Observe an expanded view of the AM USB and message signals.

Press return to continue.

Plot 21: Observe the AM LSB signal spectrum.

Label the following values:

**Hz value of each spectral component
amplitude of each spectral component
bandwidth in Hz**

Plot 22: Observe the AM USB signal spectrum.

Label the following values:

**Hz value of each spectral component
amplitude of each spectral component
bandwidth in Hz**

C. Observe the recovery of the AM SSB signals

The three-step recovery process is repeated to recover the multi-tone AM SSB signals. The ideal lowpass filter has a cutoff frequency of 420 Hz.

Press return to continue.

Plot 23: Observe the demodulated AM LSB signal (prior to filtering).

Plot 24: Observe the AM LSB recovered signal plotted over the message signal.

Press return to continue.

Plot 24: Observe the recovered amplified AM LSB signal plotted over the message signal.

Press return to continue.

Plot 25: Observe the demodulated AM USB signal (prior to filtering).

Plot 26: Observe the AM USB recovered signal plotted over the message signal.

Press return to continue.

Plot 26: Observe the recovered amplified AM USB signal plotted over the message signal.

Name: _____

Section: _____

EO 3513 Computer-aided Laboratory 7

Conventional Amplitude Modulation

(Conventional AM)

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

The m-file for this laboratory is "lab7scr.m." It may be helpful to print the m-file for reference before running the script.

Part 1—Observe the conventional amplitude modulation (conventional AM) process using a single-tone input

A. Generating the signal and spectrum

You will be observing the conventional process on the following single-tone message signal:

```
t=0:0.0001:1;           %time vector to one second
s=cos(2*pi*150*t);       %single-tone signal
```

Parseval's theorem states that average signal power can be calculated in either the time or the frequency domain. The following formula applies to calculation in the time domain:

$$P = A_0^2 + \frac{1}{2} \sum_{N=1}^{\infty} (A_N^2 + B_N^2)$$

Peak power is calculated as follows:

$$P_P = \frac{A_P^2}{2}$$

Question 1: Calculate the following values for the single-tone message signal:

**peak power
average power
baseband bandwidth**

Run the script "lab7scr.m."

Plot 1: Observe the message signal.

"Spectral.m" is used to generate spectra.

Plot 2: Observe the spectrum of the message signal.

Label the following values for the message signal:

**Hz value of the spectral component
amplitude of the spectral component
baseband signal bandwidth in Hz**

B. Observe the conventional AM signal and spectrum

The function "conv_am.m" is used to normalize the message signal and modulate it with a carrier frequency of 2000 Hz and a modulation index of 0.8. Notice that this message signal has an amplitude of 1 (normalization unnecessary).

Peak power for the single-tone conventional AM signal is calculated as follows:

$$P_p = (1 + m)^2 P_c$$

where P_c is the average power of the carrier.

Average power for the single-tone conventional AM signal is calculated as follows:

$$P = \left(1 + \frac{m^2}{2}\right) P_c$$

where P_c is the average power of the carrier.

Question 2: Predict the following values for the single-tone conventional AM signal:

**peak power
average power
bandwidth**

Press return to continue.

Plot 3: Observe the plot of the conventional AM signal.

Amplitude of the spectral components in the sidebands of the conventional AM signal can be calculated as

$$mA/2$$

where m is the modulation index and A is the amplitude of the signal tone. The amplitude of the spectral component representing the carrier is equal to the amplitude of the carrier.

Plot 4: Observe the conventional AM signal spectrum.

Label the following values:

**amplitude of each spectral component
Hz value of each spectral component
bandwidth in Hz**

C. Verify the power and bandwidth of the conventional AM signal

Average power was calculated in the time domain; it will be verified in the frequency domain. The function "psd.m" is used to generate the power spectral density of the modulated signal. The power levels associated with each frequency are added using the "sum" command to produce average signal power.

Press return to continue.

Plot 5: Observe the single-tone conventional AM power spectral density.

Question 3: From Plot 5, obtain the values representing peak and average power for the single-tone conventional AM signal, and record them.

Do your calculations for bandwidth and power agree with the computer-generated values and spectrum?

The power contained in the carrier of a conventional AM signal is sometimes referred to as "wasted" because it conveys no information from sender to receiver.

Question 4: Refer to Plot 5 and estimate the percentage of power contained in the carrier. What might be an advantage of having this amount of power transmitted in the carrier as opposed to transmission in the sidebands?

D. Observe the recovery and detection of the single-tone conventional AM signal

Conventional AM signals are recovered via use of a bandpass filter and envelope detector.

The function "recoverm.m" is used to bandpass-filter the signal between the frequencies of 1800 and 2200 Hz. Filtering is followed by envelope detection. The function "envelope.m" detects the complex magnitude of the conventional AM signal, producing a highly accurate envelope detection.

Press return to continue.

Plot 6: Observe an expanded view of the envelope-detected signal plotted over the conventional AM signal.

The DC value is subtracted from the detected signal, which is then divided by the modulation index "m."

Plot 7: Observe an expanded view of the modified envelope-detected signal.

Press return to view the message signal.

Part 2—Observe the effect of overmodulating the conventional AM signal

A. Observe the overmodulated conventional AM signal

The function "conv_am.m" is used to overmodulate the single-tone signal at a carrier frequency of 2000 Hz and a modulation index of 1.5.

Press return to continue.

Plot 8: Observe the single-tone overmodulated conventional AM signal.

Plot 9: Observe the spectrum of the single-tone overmodulated conventional AM signal.

B. Observe the effect of overmodulation on signal recovery

Question 5: What result of overmodulation prevents the use of an envelope detector for the conventional AM signal?

Question 6: What type of detection is needed for an overmodulated conventional AM signal? Why?

The function "recoverm.m" is used to bandpass-filter the signal between the frequencies of 1800 and 2200 Hz, followed by envelope detection.

Press return to continue.

Plot 10: Observe an expanded view of the envelope-detected signal plotted over the overmodulated conventional AM signal.

The DC value is subtracted from the detected signal, which is then divided by the modulation index "m."

Plot 11: Observe an expanded view of the modified overmodulated envelope-detected signal.

Press return to view the message signal.

Label the recovered signal.

Part 3—Observe the conventional AM process using a multi-tone input

A. Generating the signal and spectrum

Next you will be observing the conventional AM process on the following multi-tone message signal:

```
s=5*cos(2*pi*100*t)+4*cos(2*pi*300*t)+3*cos(2*pi*400*t); %multi-tone signal
```

Press return to continue.

Plot 12: Observe the message signal.

Plot 13: Observe the spectrum of the message signal.

Label the following values for the message signal:

Hz value of each spectral component
baseband bandwidth in Hz

B. Observe the multi-tone conventional AM signal and spectrum

The function "conv_am.m" is used to normalize the message signal and modulate it with a carrier frequency of 2000 Hz and a modulation index of 0.5.

Press return to continue.

Plot 14: Observe the plot of the multi-tone conventional AM signal.

Plot 15: Observe the spectrum of the multi-tone conventional AM signal.

Label the following variables:

**Hz value of each spectral component
bandwidth in Hz**

C. Observe the recovery and detection of the multi-tone conventional AM signal

The conventional AM signal is bandpass-filtered between the frequencies of 1500 and 2500 Hz using the function "recoverm.m," followed by envelope detection using "envelope.m."

Press return to continue.

Plot 16: Observe an expanded view of the envelope-detected signal plotted over the multi-tone conventional AM signal.

The DC value is subtracted from the detected signal, which is then divided by the modulation index "m."

Plot 17: Observe an expanded view of the modified envelope-detected multi-tone signal.

Press return to view the message signal.

Name: _____

Section: _____

EO 3513 Computer-aided Laboratory 8

Frequency Modulation (FM)

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

The m-file for this laboratory is "lab8scr.m." It may be helpful to print the m-file for reference before running the script.

Part 1--Observe the FM modulation process for single-tone input

A. Calculate theoretical average power, peak power, and bandwidth for the single-tone message signal

You will be observing the frequency modulation (FM) process on the following signal:

```
t=0:0.0001:1;      %time vector to 1 second
s=15*cos(2*pi*50*t); %single-tone signal
```

Parseval's theorem states that average signal power can be calculated in either the time or the frequency domain. The following formula applies to calculation in the time domain:

$$P = A_0^2 + \frac{1}{2} \sum_{N=1}^{\infty} (A_N^2 + B_N^2)$$

Peak power is calculated as follows:

$$P_P = \frac{A_P^2}{2}$$

Question 1: Calculate the following values for the single-tone message signal:

peak power
average power
baseband bandwidth

Computer-aided Laboratory 8—page 1

B. Observe the single-tone message signal and its spectrum

Run the script "lab8scr.m."

Plot 1: Observe the single-tone message signal.

"Spectral.m" is used to generate one-sided spectra.

Plot 2: Observe the spectrum of the single-tone message signal.

Label the following values for the message signal:

Hz value of the spectral component
amplitude of the spectral component
baseband signal bandwidth in Hz

C. Observe the process of FM modulation

The function "fm_mod.m" is used to frequency modulate the message signal at a carrier frequency of 1000 Hz and β (beta) equal to 1. (When β is specified, Δf is returned, and vice versa.) The amplitude of the returned FM signal is set at 15.

Peak power is calculated as before:

$$P_p = \frac{A_p^2}{2}$$

The average power of the FM signal is calculated as follows:

$$P = A^2 / 2$$

Recall that β and Δf are related in that $\beta f_m = \Delta f$. There are three cases for estimating transmission bandwidth, depending on the value of β :

for $\beta \leq 0.25$	$B_T \approx 2 f_m$	(narrowband FM)
for $2.5 \leq \beta < 10$	$B_T \approx 2 (1 + \beta) f_m$	(Carson's rule)
for $\beta \geq 10$	$B_T \approx 2 \beta f_m$	(wideband FM)

where f_m is the frequency of the message signal.

Question 2: Predict the following values for the single-tone FM signal:

peak power
average power
maximum frequency deviation Δf
transmission bandwidth

Press return to continue.

Plot 3: Observe an expanded view of the FM signal plotted over the message signal. Notice the frequency behavior of the FM signal at high and low amplitudes of the message signal.

D. Observe the spectrum of the FM signal

Press return to continue.

Plot 4: Observe the spectrum of the single-tone FM signal.

Label the carrier frequency and the transmission bandwidth.

Label Δf to each side of the carrier frequency.

Question 3: What is the distance between the sidebands in the FM spectrum shown in Plot 4?

E. Verify the power and bandwidth of the FM signal

Average power was calculated in the time domain; it will be verified in the frequency domain. The function "psd.m" is used to generate the power spectral density of the modulated signal. The power levels associated with each frequency are added using the "sum" command to produce average signal power.

Plot 5: Observe the single-tone power spectral density.

Question 4: From Plot 5, obtain the values representing peak and average power.

Do your theoretical calculations for bandwidth and power agree with the computer-generated values?

Question 5: Consult a table of values for Bessel functions (or use the MATLAB "bessel" function). Calculate the amplitude for the spectral components shown in the FM spectrum in Plot 4 for $n = 0$ through 6. List each frequency by its Hz value and sideband number n . Values should be consistent with the amplitudes shown for power spectral density in Plot 5.

F. Control the bandwidth of the FM signal by varying β

The bandwidth of an FM signal can be controlled by fixing either β or Δf . The single-tone message signal will be frequency modulated four times with a carrier frequency of 2500 Hz, using the following four values of β : 0.1, 1, 5, and 20.

Question 6: Calculate the maximum frequency deviation Δf associated with each of the four values of β :

0.1
1
5
20

Question 7: Predict the transmission bandwidth for each of the FM signals referred to in Question 6.

Press return to continue.

Plot 6: Observe the spectrum of the FM signal for $\beta = 0.1$.

Label the spectrum with the transmission bandwidth calculated in Question 7.

Plot 7: Observe the spectrum of the FM signal for $\beta = 1$.

Label the spectrum with the transmission bandwidth calculated in Question 7.

Press return to continue.

Plot 8: Observe the spectrum of the FM signal for $\beta = 5$.

Label the spectrum with the transmission bandwidth calculated in Question 7.

Plot 9: Observe the spectrum of the FM signal for $\beta = 10$.

Label the spectrum with the transmission bandwidth calculated in Question 7.

Part 2--Observe the FM modulation process for multi-tone input

A. Calculate theoretical average power, peak power, and bandwidth for the multi-tone message signal

You will next be observing the frequency modulation (FM) process on the following signal:

$s=8*\cos(2*\pi*75*t)+12*\cos(2*\pi*25*t);$ %multi-tone signal

Question 8: Calculate the following values for the multi-tone message signal:

**peak power
average power
baseband bandwidth**

B. Observe the multi-tone message signal and its spectrum

Plot 10: Observe the multi-tone message signal.

Plot 11: Observe the spectrum of the multi-tone message signal.

Label the following values for the message signal:

**Hz value of the spectral components
amplitude of the spectral components
baseband signal bandwidth in Hz**

C. Observe the process of FM modulation

The function "fm_mod.m" is used to frequency modulate the message signal at a carrier frequency of 1500 Hz and β (beta) equal to 1. The amplitude of the returned FM signal is set at 20. When working with multi-tone signals, f_m is considered to be equal to the highest frequency in the message signal.

Question 9: Predict the following values for the multi-tone FM signal:

**peak power
average power
maximum frequency deviation Δf
transmission bandwidth**

Press return to continue.

Plot 12: Observe an expanded view of the FM signal plotted over the message signal. Notice the frequency behavior of the FM signal at high and low amplitudes of the message signal.

D. Observe the spectrum of the FM signal

Press return to continue.

Plot 13: Observe the spectrum of the multi-tone FM signal.

Label the carrier frequency and the transmission bandwidth.

Question 10: What is the distance between the sidebands in the FM spectrum shown in Plot 13?

E. Verify the power and bandwidth of the FM signal

Average power is verified in the frequency domain using the function "psd.m" to generate the power spectral density of the modulated signal. The power levels associated with each frequency are added using the "sum" command to produce average signal power.

Plot 14: Observe the multi-tone power spectral density.

Question 11: From Plot 14, obtain the values representing peak and average power.

Do your theoretical calculations for bandwidth and power agree with the computer-generated values?

F. Control the bandwidth of the FM signal by varying Δf

The multi-tone message signal will be frequency modulated four times with a carrier frequency of 2500 Hz, using the following four values of Δf : 25, 100, 500, and 1000.

Question 12: Calculate the value of β associated with each of the four values of Δf :

25
100
500
1000

Question 13: Predict the transmission bandwidth for each of the FM signals referred to in Question 12.

Press return to continue.

Plot 15: Observe the spectrum of the FM signal for $\Delta f = 25$.
 Label the spectrum with the transmission bandwidth calculated
 in Question 13.

Plot 16: Observe the spectrum of the FM signal for $\Delta f = 100$.
 Label the spectrum with the transmission bandwidth calculated
 in Question 13.

Press return to continue.

Plot 17: Observe the spectrum of the FM signal for $\Delta f = 500$.
 Label the spectrum with the transmission bandwidth calculated
 in Question 13.

Plot 18: Observe the spectrum of the FM signal for $\Delta f = 1000$.
 Label the spectrum with the transmission bandwidth calculated
 in Question 13.

**This page is intentionally
left blank.**

Name: _____

Section: _____

EO 3513 Computer-aided Laboratory 9 *Radio Frequency Digital Modulation Methods* *(ASK, FSK, BPSK, and QPSK)*

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

The m-file for this laboratory is "lab9scr.m." It may be helpful to print the m-file for reference before running the script.

Part 1—Observe the process of amplitude shift keying (ASK) and coherent detection

A. Generating the digital message signal

Digital signals in this laboratory are generated using the "random" command in MATLAB; four new digital signals are generated each time the script is run.

ASK signals require a unipolar digital signal. The function "nrzluni.m" is used to generate a NRZL unipolar digital signal at a bit rate of 100 bits per second.

Question 1: Calculate the bit duration τ for this signal.

From the command window, obtain the values of the first 10 bits in the bitstream. Record these values.

Run the script "lab9scr.m."

Plot 1: Observe the NRZL unipolar digital message signal, with a bit rate of 100 bits per second.

Label the values (0 or 1) of the first 10 bits, and the bit duration τ .

"Spectral.m" is used to generate one-sided spectra. Recall that a "coarse" approximation for baseband bandwidth of a digital signal is $0.5/\tau$.

Question 2: Calculate the approximate baseband bandwidth of the NRZL unipolar digital message signal.

Plot 2: Observe the spectrum of the NRZL unipolar digital message signal.

Label the baseband bandwidth in Hz.

B. Generating the ASK signal

The digital message signal is modulated by a cosine with a carrier frequency of 800 Hz.

Press return to continue.

Plot 3: Observe the ASK signal at a carrier frequency of 800 Hz.

Label the values (0 or 1) of the bits shown.

Question 3: Why is ASK modulation often referred to as “on-off keying”?

Plot 4: Observe the ASK signal spectrum.

Label the carrier frequency of the ASK signal.

C. Coherent detection of the ASK signal

The function “recover.m” is used to bandpass-filter the ASK signal between the frequencies of 700 Hz and 900 Hz. The recovered signal is then multiplied by its carrier.

Question 4: Describe a noncoherent method of detection for this ASK signal. Why will this method work for ASK?

Press return to continue.

Plot 5: Observe the spectrum of the demodulated ASK signal prior to filtering.

The function “recoverm.m” is used to lowpass filter the signal at 100 Hz. The recovered signal is then multiplied by a factor of 2 to restore its amplitude.

Plot 6: Observe the message signal plotted over the recovered ASK signal.

Press return to observe the amplified recovered ASK signal.

Part 2—Observe the process of frequency shift keying (FSK) and coherent detection

A. Generating the digital message signal

The FSK signal can be based on either a unipolar or bipolar digital signal. The function "nrzluni.m" is used to generate a unipolar digital message signal from a random bitstream, at a bit rate of 100 bits per second.

Question 5: From the command window, obtain the values of the first 10 bits in the bitstream. Record these values.

Press return to continue.

Plot 7: Observe the NRZL unipolar digital message signal, with a bit rate of 100 bits per second.

Label the values (0 or 1) of the first 10 bits, and the bit duration τ .

Plot 8: Observe the spectrum of the NRZL unipolar digital message signal.

Label the baseband bandwidth in Hz.

B. Generating the FSK signal

The FSK signal is generated using the function "fsk.m." Within this function, the bits representing 1's are modulated at a frequency of 1500 Hz, and the bits representing 0's are modulated at a frequency of 500 Hz.

Press return to continue.

Plot 9: Observe the FSK signal at a carrier frequencies of 500 and 1500 Hz.

Label the values (0 or 1) of the bits shown.

Plot 10: Observe the FSK signal spectrum.

Label the two carrier frequencies of the FSK signal.

C. Coherent detection of the FSK signal

The function "recoverm.m" is used to bandpass filter the FSK signal between the frequencies of 400 Hz and 1600 Hz. The recovered signal is multiplied twice, once by each carrier signal.

Press return to continue.

Plot 11: **Observe the spectrum of the higher-frequency demodulated signal, prior to filtering.**

Press return to observe the spectrum of the lower-frequency demodulated signal, prior to filtering.

The two signals are combined in the time domain by subtracting the lower frequency signal from the higher frequency signal.

Plot 12: **Observe the spectrum of the combined upper and lower frequency signals, prior to filtering.**

"Recoverm.m" is used to lowpass-filter the combined signal at 100 Hz. Notice that the 1's in the message signal are now represented by a positive value and the 0's are represented by a negative value. The recovered signal is then multiplied by a factor of 2 to restore its amplitude.

Press return to continue.

Plot 13: **Observe the recovered FSK signal plotted over the message signal.**

Press return to observe the amplified recovered FSK signal.

Part 3—Observe the process of binary phase shift keying (BPSK) and coherent detection

A. Generating the digital message signal

The BPSK signal is based on a bipolar digital signal. The function "nrzlb.m" is used to generate a bipolar digital message signal from a random bitstream, at a bit rate of 100 bits per second.

Question 6: **From the command window, obtain the values of the first 10 bits in the bitstream. Record these values.**

Press return to continue.

Plot 14: **Observe the NRZL bipolar digital message signal, with a bit rate of 100 bits per second.**

Label the values (0 or 1) of the first 10 bits, and the bit duration τ .

Plot 15: **Observe the spectrum of the NRZL bipolar digital message signal.**

Label the baseband bandwidth in Hz.

B. Generating the BPSK signal

The BPSK signal is generated by modulating the message signal with a cosine at a carrier frequency of 800 Hz.

Press return to continue.

Plot 16: **Observe the BPSK signal at a carrier frequency of 800 Hz. Notice the phase shifts in the signal.**

Press return to observe the message signal.

Label the values (0 or 1) of the bits shown.

Plot 17: **Observe the BPSK signal spectrum.**

Label the carrier frequency of the BPSK signal.

C. Coherent detection of the BPSK signal

The first step in coherent detection of the BPSK signal is to square the signal in the time domain.

Press return to continue.

Plot 18: **Observe the squared BPSK signal spectrum.**

Question 7: **What are the effects in the frequency domain of squaring the BPSK signal?**

The signal frequency is now twice the frequency desired. The function "freq_div.m" is used to shift the frequency of the squared BPSK signal back to the carrier frequency of 800 Hz.

Plot 19: **Observe the frequency-divided BPSK signal spectrum.**

"Recoverm.m" is used to lowpass-filter the signal at 1000 Hz. In the time domain, the recovered signal is multiplied by the received signal.

Press return to continue.

Plot 20: **Observe the spectrum of the recovered demodulated BPSK signal, prior to filtering.**

“Recoverm.m” is now used to lowpass-filter the signal at 100 Hz. The recovered signal, which has a much higher amplitude than its message signal, is normalized by dividing it by its maximum amplitude.

Plot 21: **Observe the message signal plotted over the recovered normalized signal.**

Part 4—Observe the process of quadriphase shift keying (QPSK) and coherent detection

A. Generating the digital message signal

The QPSK signal is based on a bipolar digital signal. The function “nrzlb.m” is used to generate a bipolar digital message signal from a random bitstream, at a bit rate of 100 bits per second. (The first 8 bits are established in a pattern that will demonstrate the four types of phase shifts present in the QPSK signal.)

Question 8: From the command window, obtain the values of the first 10 bits in the bitstream. Record these values.

Press return to continue.

Plot 22: **Observe the NRZL bipolar digital message signal, with a bit rate of 100 bits per second.**

Label the values (0 or 1) of the first 10 bits, and the bit duration τ .

Plot 23: **Observe the spectrum of the NRZL bipolar digital message signal.**

Label the baseband bandwidth in Hz.

B. Generating the QPSK signal

The first step in generating the QPSK signal is to split the signal by putting it through a serial-to-parallel converter. One of the output signals is composed of the odd bits, the other of the even bits. The bits in each output signal have a bit rate of half that of the input signal.

Press return to continue.

Plot 24: Observe the signal composed of odd bits.

Label the values (0 or 1) of the first 5 bits shown, and the bit duration τ .

Plot 25: Observe the signal composed of even bits.

Label the values (0 or 1) of the first 5 bits shown, and the bit duration τ .

Next, the signal composed of odd bits is modulated by a positive cosine with a carrier frequency of 400 Hz. The signal composed of even bits is modulated by a negative sine with a carrier frequency of 400 Hz.

Press return to continue.

Plot 26: Observe the modulated "odd-bit" signal. Notice the phase shifts present.

Press return to observe the "odd-bit" signal.

Plot 27: Observe the modulated "even-bit" signal. Notice the phase shifts present.

Press return to observe the "even-bit" signal.

The QPSK modulation process is completed by summing the two modulated signals in the time domain.

Press return to continue.

Plot 28: Observe the QPSK signal.

Label the phase shifts present in the signal.

Plot 29: Observe the QPSK spectrum.

Label the carrier frequency of the QPSK signal.

C. Coherent detection of the QPSK signal

The first step in coherent detection of the QPSK signal is to generate two signals by multiplying (demodulating) the QPSK signal in the time domain. Multiplication by a positive cosine function at the carrier frequency of 400 Hz creates the "upper" odd-bit signal. Multiplication by a negative sine function at the carrier frequency creates the "lower" even-bit signal..

Press return to continue.

Plot 30: Observe the demodulated "upper" signal.

Press return to observe the demodulated "lower" signal.

Plot 31: Observe the demodulated "upper" signal spectrum, prior to filtering.

Press return to observe the demodulated "lower" signal spectrum, prior to filtering.

"Recoverm.m" is used to separately lowpass-filter and recover each signal, using a cutoff frequency of 100 Hz.

Press return to continue.

Plot 32: Observe the odd-bit signal plotted over the recovered "upper" signal.

Plot 33: Observe the even-bit signal plotted over the recovered "lower" signal.

The final step in coherent QPSK detection is to join the signals using a parallel-to-serial converter, performed by the function "par_ser.m." The bit rate of the output signal is twice the bit rate of each of the input signals.

Press return to continue.

Plot 34: Observe the digital message signal plotted over the combined "upper" and "lower" recovered signals.

Question 9: What is the chief advantage of quadriphase shift keying over bipolar phase shift keying?

APPENDIX B—COMPUTER-AIDED LABORATORY KEYS

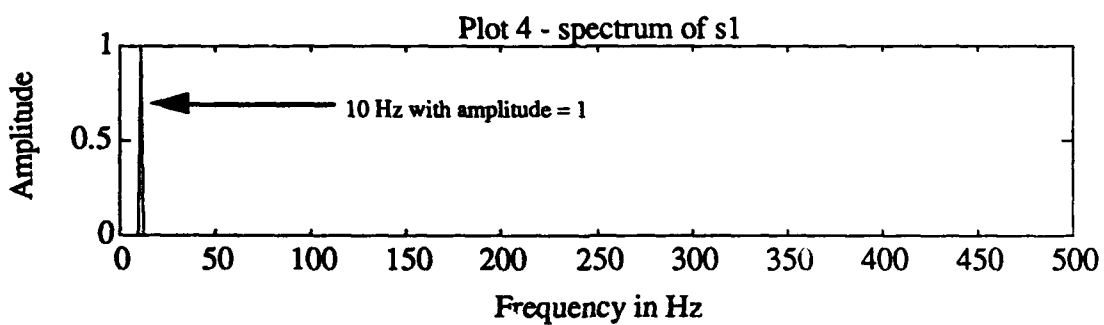
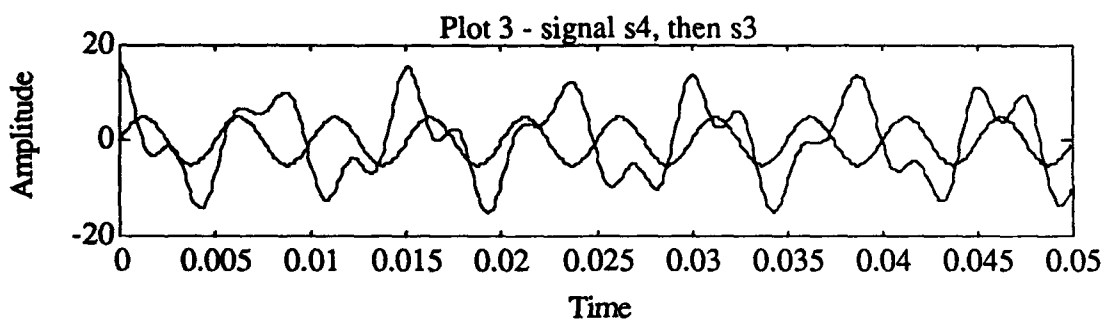
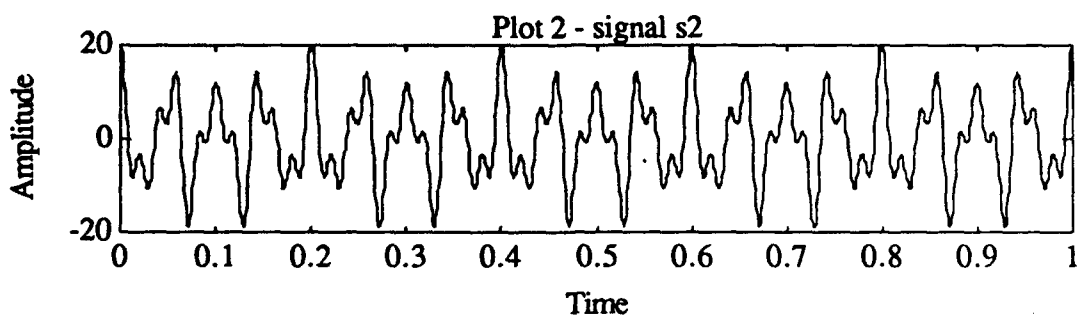
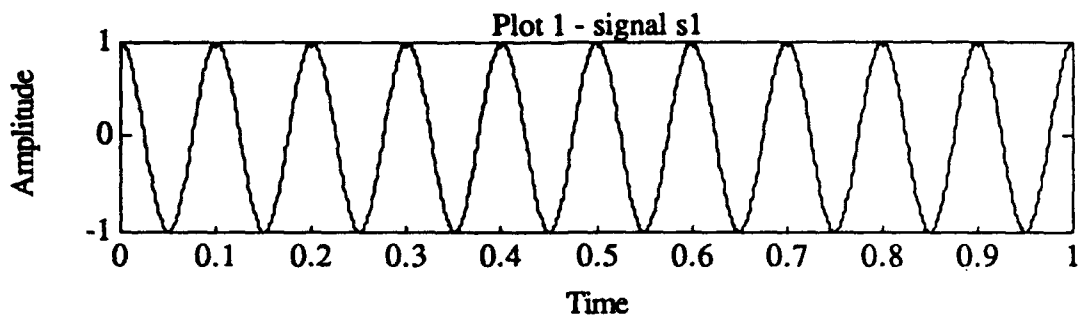
EO 3513 Computer-aided Laboratory 1 Key *Signal and Spectrum Generation*

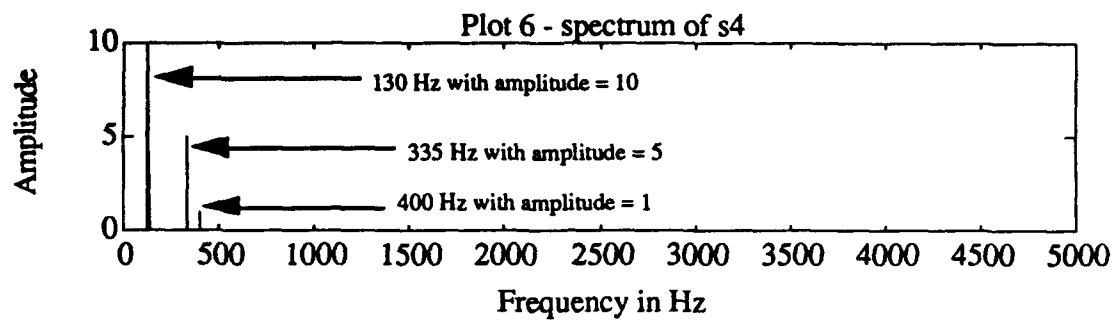
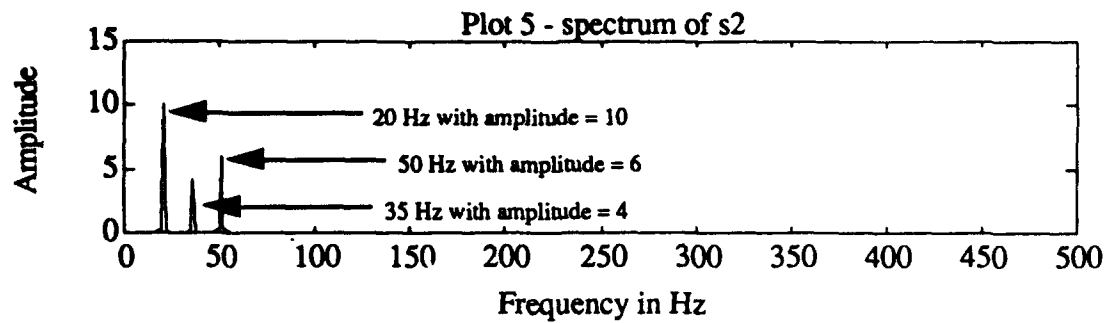
Question 1: Compare Plots 5 and 6. Why does Plot 6 display more frequencies than Plot 5?

Answer: For signal s2 shown in Plot 5, the step size of 0.001 produced a one-sided spectrum of only 500 Hz; for signal s4 shown in Plot 6, the step size of 0.0001 produced a one-sided spectrum of 5000 Hz.

Question 2: Given the spectral plot of a multi-tone signal, how could you determine the amplitudes and frequencies of each of the signal tones?

Answer: On a one-sided spectrum, the amplitude of each signal tone is plotted against frequency in Hz. The amplitude of each signal tone could be found by observing the amplitude of the spectral component. Frequencies could be determined by observing the Hz values of the spectral components.





lab1scr.m

%Computer-aided Lab 1 script for student use

%%%

%Computer-aided Lab 1 Signal and Spectrum Generation

%%%

%Part 1--Observe signal generation

%A. Establish a time vector

clear

clg

t1=0:.001:1; %time vector

%B. Generate a signal

s1=cos(2*pi*10*t1); %single-tone signal

%multi-tone signal

s2=10*cos(2*pi*20*t1)+4*cos(2*pi*35*t1)+6*cos(2*pi*50*t1);

%C. Controlling signal plots

%Plot 1

subplot(211),

plot(t1,s1)

title('Plot 1 - signal s1')

xlabel('Time')

ylabel('Amplitude')

%Plot 2

plot(t1,s2)

title('Plot 2 - signal s2')

xlabel('Time')

ylabel('Amplitude')

pause

clg

t2=0:.0001:1; %time vector

s3=5*sin(2*pi*200*t2); %single-tone signal

%multi-tone signal

s4=10*cos(2*pi*130*t2)+5*cos(2*pi*335*t2)+cos(2*pi*400*t2);

%Plot 3

Computer-aided Laboratory 1 Key—page 4

```

subplot(211),
plot(t2(1:500),s4(1:500))
title('Plot 3 - signal s4, then s3')
xlabel('Time')
ylabel('Amplitude')
hold on
pause
plot(t2(1:500),s3(1:500),'b')
hold off

pause

%%%%%%%%%%%%%%
%Part 2--Observe spectrum generation
%A. Calling a function

[spec1,shortHz]=spectral(s1,.001); %generate spectrum for s2

%Plot 4

subplot(212),
plot(shortHz,spec1) %plot spectrum of s1
title('Plot 4 - spectrum of s1')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

spec2=spectral(s2,.001); %generate spectrum for s2

%Plot 5

subplot(211),
plot(shortHz,spec2) %plot spectrum of s2
title('Plot 5 - spectrum of s2')
xlabel('Frequency in Hz')
ylabel('Amplitude')

[spec4,longHz]=spectral(s4,.0001); %generate spectrum for s4

%Plot 6

subplot(212),
plot(longHz,spec4) %plot spectrum of s4
title('Plot 6 - spectrum of s4')
xlabel('Frequency in Hz')
ylabel('Amplitude')

```

**This page is intentionally
left blank.**

EO 3513 Computer-aided Laboratory 2 Key

Sampling and Recovery

Question 1: Calculate the following values, in seconds, for the naturally- and flattop-sampled signals:

**sampling period T
pulse duration τ**

Answer: $T = 1/f_s \Rightarrow 1/1000 \Rightarrow 0.001$ seconds
 $\tau = d * T \Rightarrow 0.5 * 0.001 \Rightarrow 0.0005$ seconds

Question 2: Describe the distinguishing pulse shape of each sampled signal:

**naturally-sampled
flattop-sampled
impulse-sampled**

Answer: naturally-sampled - pulses follow the shape of the message signal
flattop-sampled - pulses have the amplitude of the message signal at the pulse beginning, but remain flat over pulse duration
impulse-sampled - pulses are ideal impulses with the amplitude of the signal

Question 3: Describe the overall shape of each spectrum. Does each of the spectral plots conform to your theoretical expectations? Note any discrepancies.

Answer: The naturally-sampled signal spectrum consists of groups of frequencies which have a "sinc" shape to their envelope. The impulse-sampled signal spectrum shows frequencies which have constant amplitudes. The flattop-sampled signal spectrum shows frequencies that individually conform to the "sinc" envelope.

Naturally-sampled and flattop-sampled spectra are as expected, but the amplitude of the spectral components in the impulse-sampled signal spectrum should remain constant, not decline (due to the computer's inability to generate a perfect impulse).

Question 4: Calculate P_N for $N = 1$, $N = 2$, and $N = 3$.

Compare with the values shown on the spectral plot for natural sampling.

Answer: For $N = 1$ $P_N = 0.3183$
For $N = 2$ $P_N = 0$
For $N = 3$ $P_N = -0.1061$

Values are consistent with those on the spectral plots. (Note that the absolute values are plotted.)

Question 5: Calculate P_N for $f = 550$, $f = 700$, and $f = 880$.

Compare with the values shown on the spectral plot for flat-top sampling.

Answer: For $f = 550$ $P_N = 0.4401$
For $f = 700$ $P_N = 0.4052$
For $f = 880$ $P_N = 0.3553$

Values are consistent with those on the spectral plots.

Question 6: What is the minimum theoretical sampling frequency for the message signal s ?

Answer: $\geq 2 * 450 \text{ Hz} \Rightarrow \geq 900 \text{ Hz}$

Question 7: Compare the undersampled signal spectra in plot 12 to its counterpart in plot 7. What is the effect of undersampling on the spectrum? What is the effect of undersampling on the signal recovery?

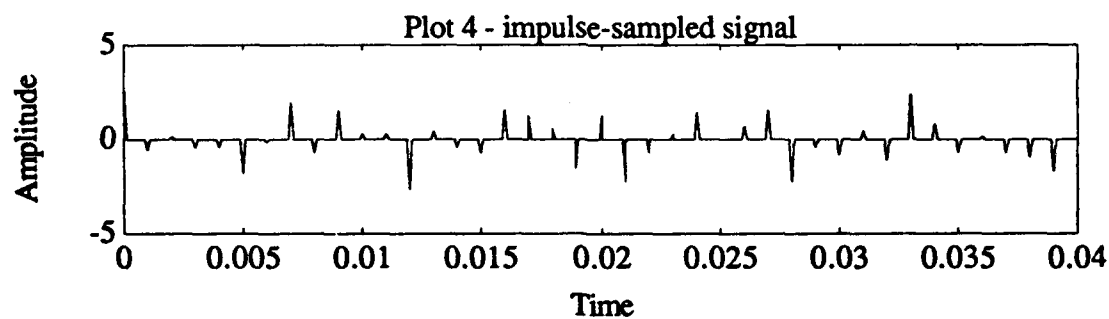
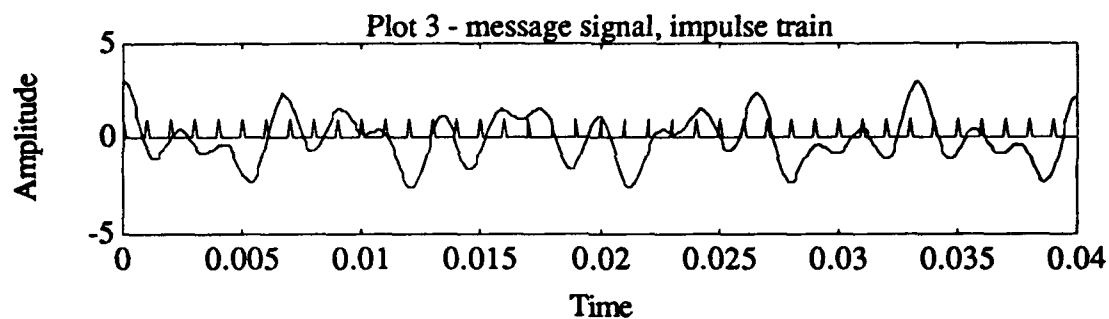
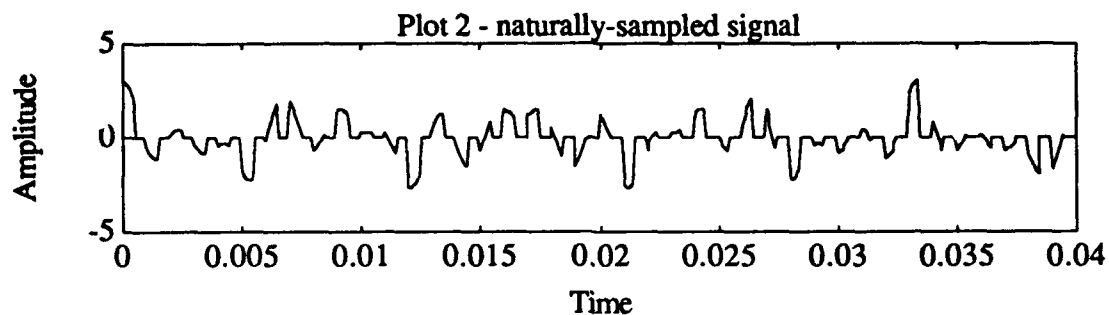
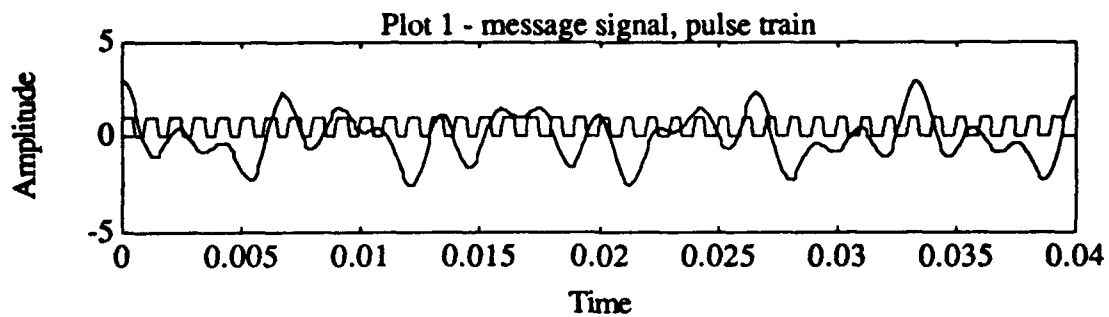
Answer: The replicas of the baseband message signal frequencies produced by sampling overlap, and prevent proper recovery of the message signal (this effect is called "aliasing").

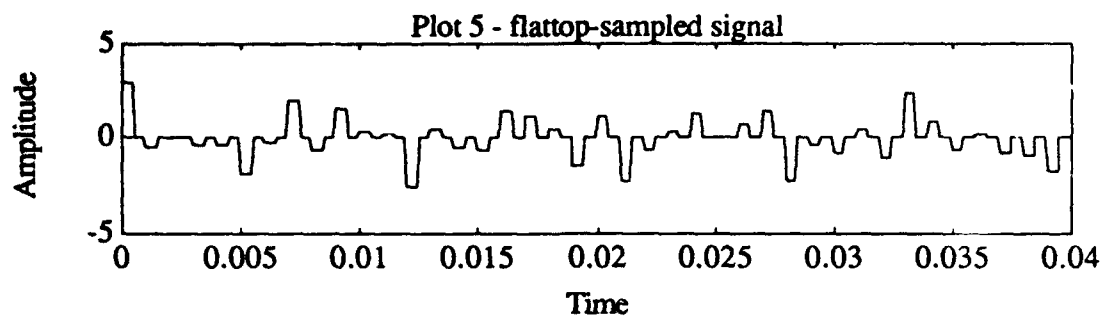
Question 8: What is the effect of reducing the duty cycle on the sampled signal baseband bandwidth?

Answer: As the pulse width decreases, the sampled signal baseband bandwidth increases, illustrating the trade-off between transmission power and required bandwidth.

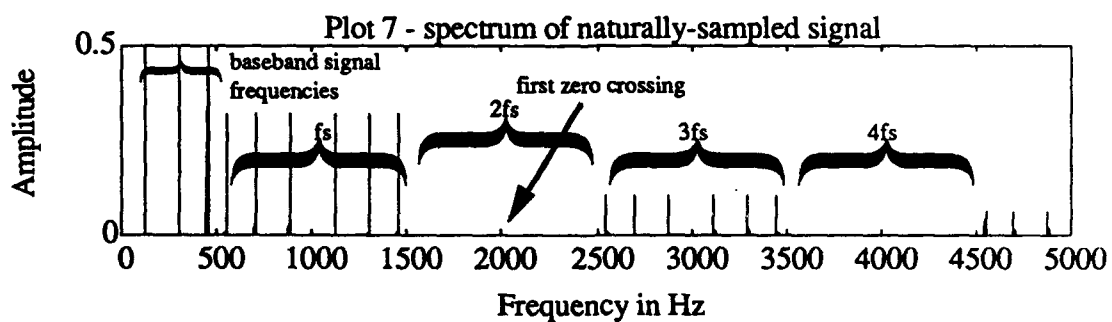
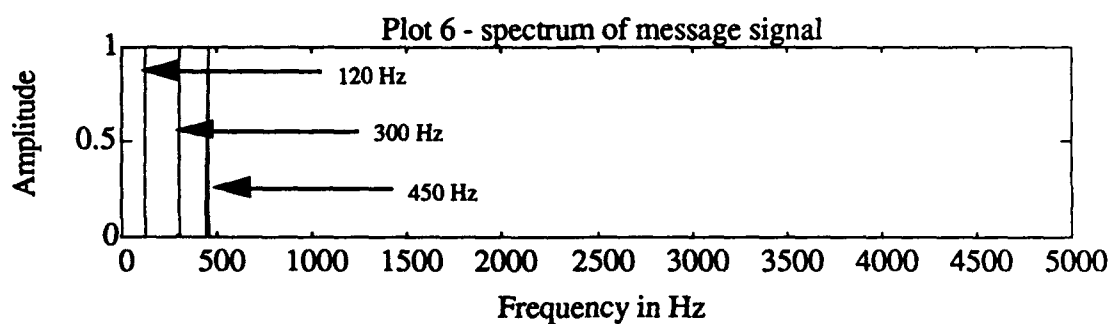
The decreased pulse width causes the pulse shapes to change more frequently; thus higher frequencies are needed to capture the changes.

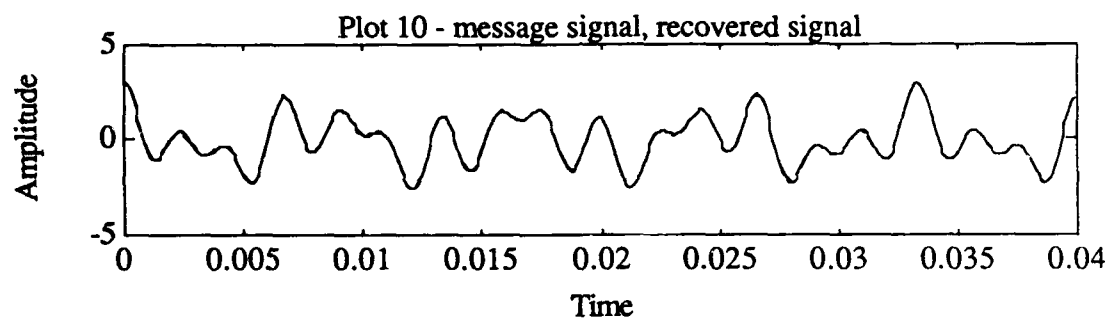
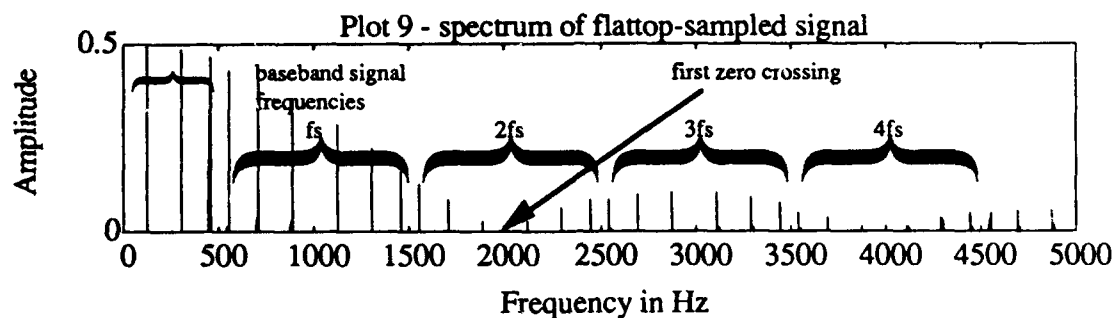
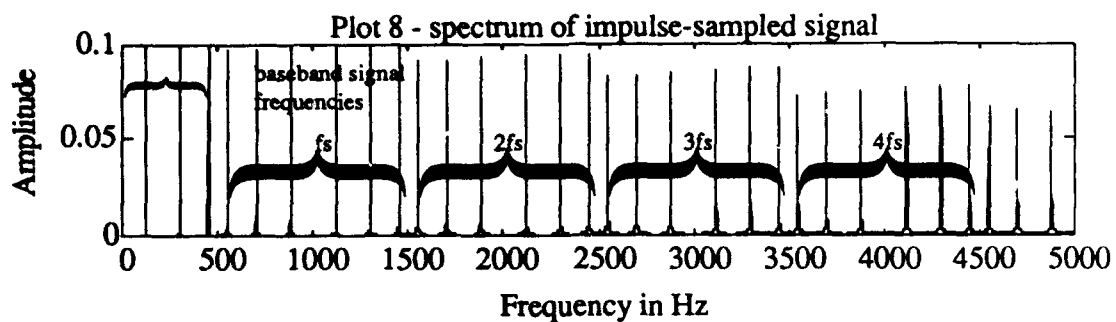
Computer-aided Laboratory 2 Key—page 2



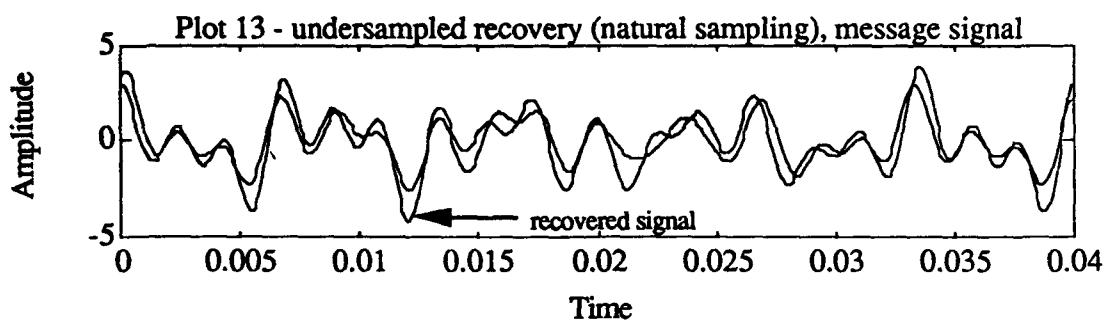
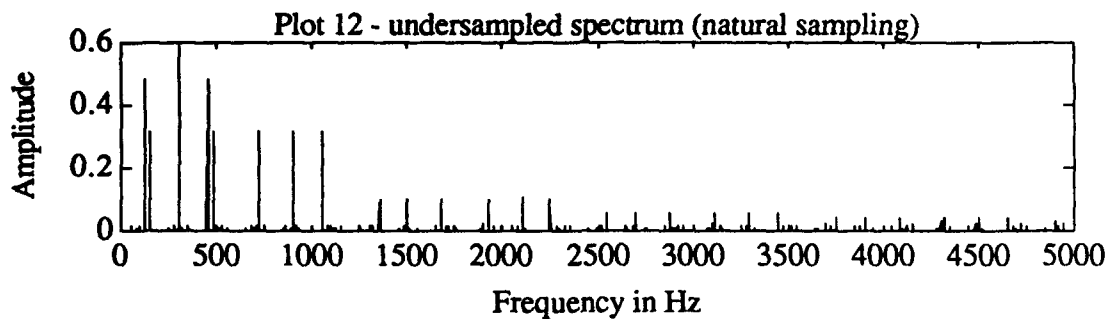
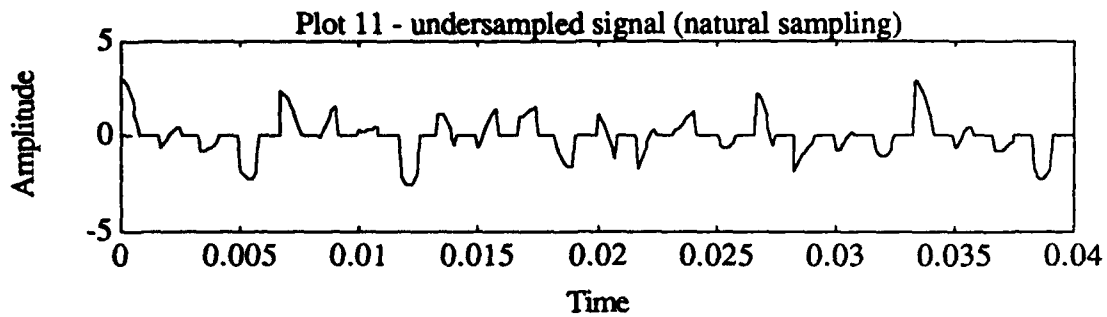


NO PLOT HERE--JUST PRESS RETURN

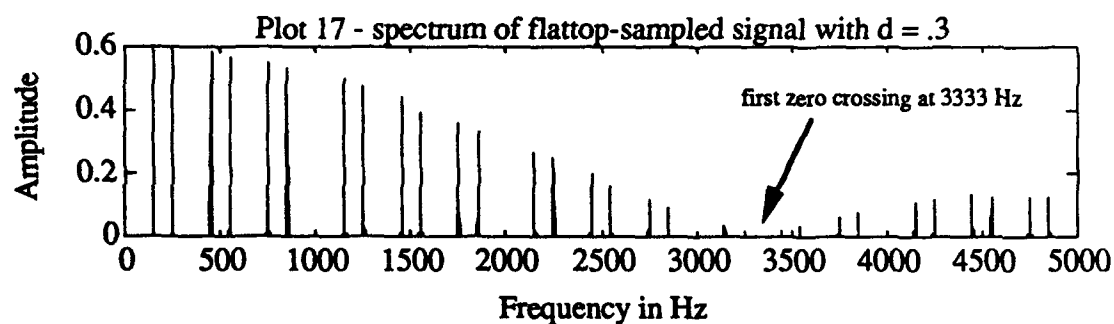
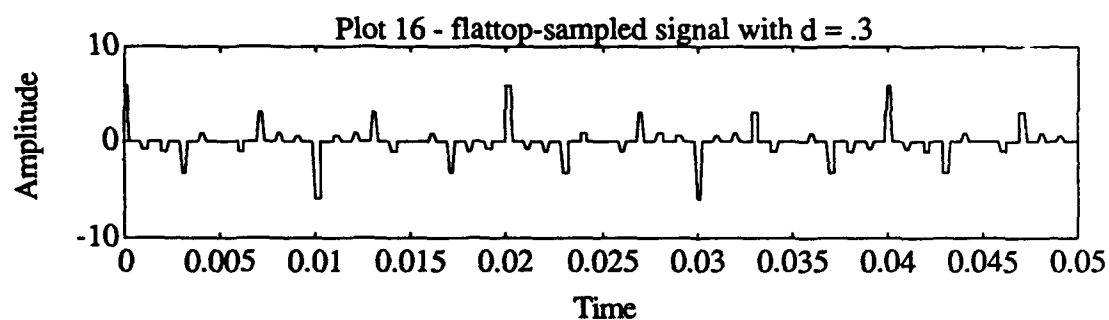
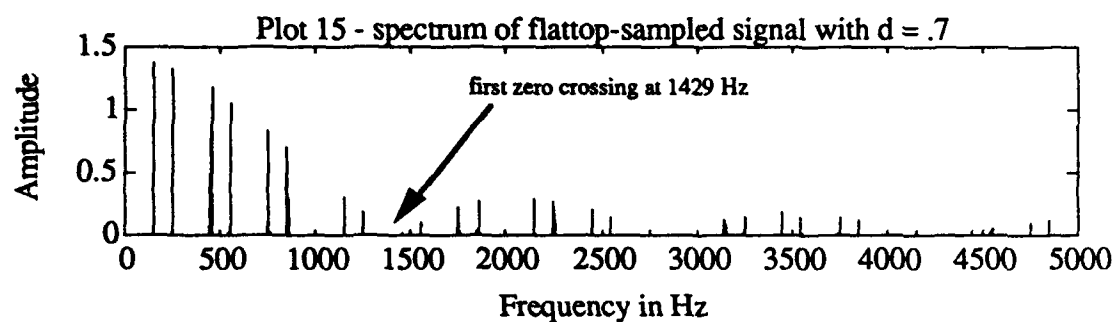
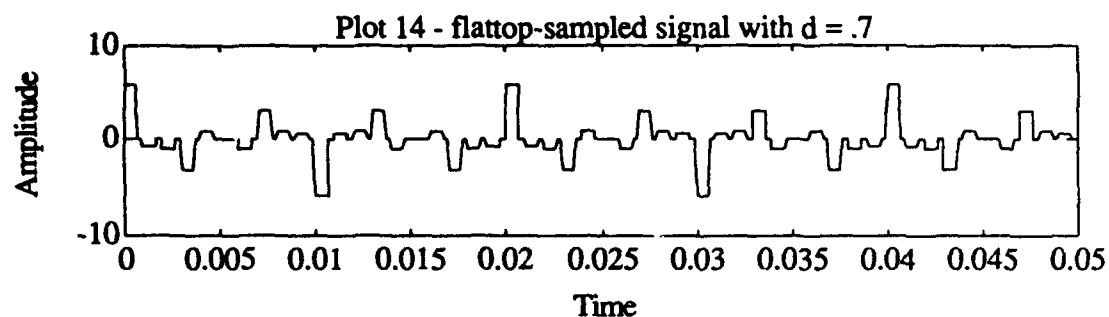




NO PLOT HERE--JUST PRESS RETURN



NO PLOT HERE--JUST PRESS RETURN



lab2scr.m

%Computer-aided Lab 2 script for student use

%%
%Computer-aided Lab 2 Sampling and Recovery
%%
%Part 1--Observe the effects of sampling on the spectrum
%A. Generating the signal

clear
clg

delta_t=.0001; %step size
t=0:delta_t:1; %time vector
%signal vector
s=cos(2*pi*120*t)+cos(2*pi*300*t)+cos(2*pi*450*t);
samprate=1000;
d=.5;

%Plot 1

subplot(211), %plot message and pulse train
plot(t(1:400),s(1:400))
title('Plot 1 - message signal, pulse train')
xlabel('Time')
ylabel('Amplitude')
hold on

pause

%naturally-sample the signal
[natsig1,pulstrn1]=natsamp(s,delta_t,samprate,d);

%B. Sampling the signal

%Plot 1 continued

plot(t(1:400),pulstrn1(1:400),'b')
hold off

%Plot 2

subplot(212), %plot naturally-sampled signal
plot(t(1:400),natsig1(1:400))
title('Plot 2 - naturally-sampled signal')
xlabel('Time')
ylabel('Amplitude')

```

pause
clg

    %impulse-sample the signal
[impsig1,imptrn]=impsamp(s,delta_t,samprate);
    %flattop-sample the signal
flatsig1=flattop(s,delta_t,samprate,d);

%Plot 3

subplot(211), %plot message and impulse train
plot(t(1:400),[s(1:400);imptrn(1:400)])
title('Plot 3 - message signal, impulse train')
xlabel('Time')
ylabel('Amplitude')

%Plot 4

subplot(212), %plot impulse-sampled signal
plot(t(1:400),impsig1(1:400))
title('Plot 4 - impulse-sampled signal')
xlabel('Time')
ylabel('Amplitude')

pause
clg

%Plot 5

subplot(211), %plot flattop-sampled signal
plot(t(1:400),flatsig1(1:400))
title('Plot 5 - flattop-sampled signal')
xlabel('Time')
ylabel('Amplitude')

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

pause
clg
clear pulstrn1;clear pulstrn2;clear imptrn;

%C. Generating the spectrum

    %generate spectrum for s
[spec_s,Hz]=spectral(s,delta_t);

%Plot 6

```



```

subplot(211),
plot(Hz,spec_s)
title('Plot 6 - spectrum of message signal')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear spec_s;

%generate spectrum for naturally sampled signal
[specnat1,H,ffinat]=spectral(natsig1,delta_t);

%Plot 7

subplot(212), %plot spectrum of naturally-sampled signal
plot(Hz,specnat1)
title('Plot 7 - spectrum of naturally-sampled signal')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg
clear specnat1;

%generate spectrum for impulse sampled signal
[specimp1,H,fftimp]=spectral(impsig1,delta_t);

%Plot 8

subplot(211),
plot(Hz,specimp1)
title('Plot 8 - spectrum of impulse-sampled signal')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear specimp1;clear impsig1;

%generate spectrum for flattop sampled signal
[specflat1,H,fftflat]=spectral(flatsig1,delta_t);

%Plot 9

subplot(212), %plot spectrum of flattop-sampled signal
plot(Hz,specflat1)
title('Plot 9 - spectrum of flattop-sampled signal')
xlabel('Frequency in Hz')
ylabel('Amplitude')

```

```

pause
clg

clear specflat1;clear flatsig1;clear fftflat;clear fftimp;

%%%%%%%%%%%%%%
%Part 2--Observe the message signal recovery
%A. Recovering the message signal

recnat1=recovers(fftmat,d,'ideallow',Hz,500);
clear fftmat;

%Plot 10

subplot(211),
plot(t(1:400),s(1:400))
title('Plot 10 - message signal, recovered signal')
xlabel('Time')
ylabel('Amplitude')
hold on
pause(3)
plot(t(1:400),recnat1(1:400),'g')
hold off

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

clear recnat1;

pause
clg

%%%%%%%%%%%%%%
%Part 3--Observe the effects of aliasing on the spectrum
% and recovery
%A. Generating undersampled signals

samprate=600;
d=.5;
            %impulse-sample the signal
natsig2=natsamp(s,delta_t,samprate,d);
            %generate the spectrum
[specnat2,Hz,fftmat2]=spectral(natsig2,delta_t);

%Plot 11

subplot(211),
plot(t(1:400),natsig2(1:400))

```

```

title('Plot 11 - undersampled signal (natural sampling)')
xlabel('Time')
ylabel('Amplitude')

%Plot 12

subplot(212),
plot(Hz,specnat2)
title('Plot 12 - undersampled spectrum (natural sampling)')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

clear specnat2;
recnat2=recover(fftmat2,d,'ideal',Hz,500);
clear fftmat2;

%Plot 13

subplot(211),
plot(t(1:400),recnat2(1:400))
title('Plot 13 - undersampled recovery (natural sampling), message signal')
xlabel('Time')
ylabel('Amplitude')
hold on
pause(3)
plot(t(1:400),s(1:400),'b')
hold off

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

pause
clg
clear

%%%%%%%%%%%%%%
%Part 4--Observe the effects of altering the duty cycle on the
%   baseband bandwidth

delta_t=.0001; %regenerate the signal variables
samprate=1000;
t=0:delta_t:1;
s=2*(cos(2*pi*150*t)+cos(2*pi*250*t)+cos(2*pi*450*t));

%A. Flattop-sample the signal using a larger duty cycle

```

```

flatsigbig=flattop(s,delta_t,samprate,.7); %sample the signal

%Plot 14

subplot(211), %plot the signal
plot(t(1:500),flatsigbig(1:500))
title('Plot 14 - flattop-sampled signal with d = .7')
xlabel('Time')
ylabel('Amplitude')

[specflatbig,HZ]=spectral(flatsigbig,delta_t); %generate the spectrum

%Plot 15

subplot(212), %plot the spectrum
plot(HZ,specflatbig)
title('Plot 15 - spectrum of flattop-sampled signal with d = .7')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
c1g

%B. Flattop-sample the signal using a smaller duty cycle

flatsigshort=flattop(s,delta_t,samprate,.3); %sample the signal

%Plot 16

subplot(211), %plot the signal
plot(t(1:500),flatsigshort(1:500))
title('Plot 16 - flattop-sampled signal with d = .3')
xlabel('Time')
ylabel('Amplitude')

specflatshort=spectral(flatsigshort,delta_t);

%Plot 17

subplot(212), %plot the spectrum
plot(HZ,specflatshort)
title('Plot 17 - spectrum of flattop-sampled signal with d = .3')
xlabel('Frequency in Hz')
ylabel('Amplitude')

```

This page is intentionally
left blank.

EO 3513 Computer-aided Laboratory 3 Key Pulse Modulation (PAM, PWM, PPM)

Question 1: Calculate the following values, in seconds, for the PAM signal:

sampling period T
pulse duration τ

Answer: $T = 1/f_s \Rightarrow 1/500 \Rightarrow 0.002$ seconds
 $\tau = d * T \Rightarrow 0.5 * .002 = 0.001$ seconds

Question 2: The maximum signal amplitude of s is 8.8, the modulation rate is 500 Hz, and the maximum pulse duration is 0.8 of the sampling period. Calculate the duration in seconds of the widest pulse that could occur for its PWM signal.

Answer: Maximum pulse duration = $0.8 * T \Rightarrow 0.8 * 0.002 \Rightarrow 0.0016$ seconds

Question 3: The zero crossings in this signal occur halfway between the minimum and maximum signal values. Calculate the pulse duration in seconds for a PWM pulse that occurs at the beginning of a zero crossing.

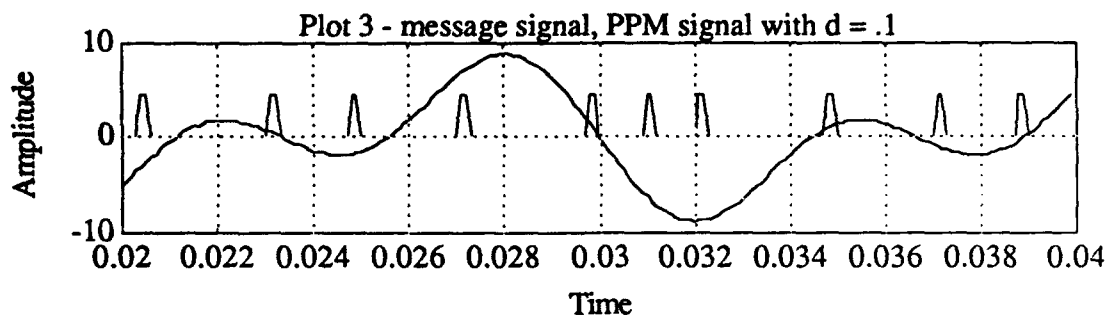
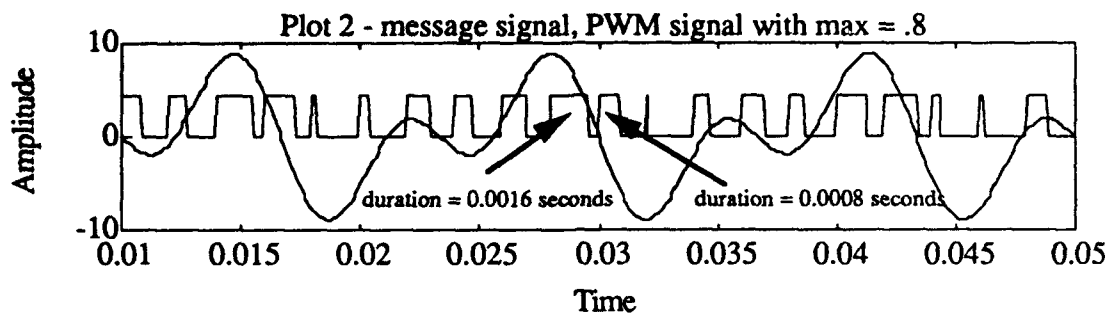
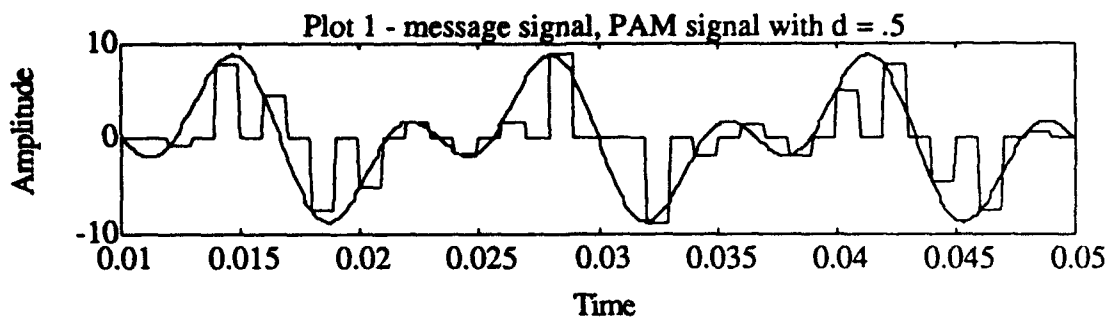
Answer: Pulse duration at zero crossing = $0.0016/2 \Rightarrow 0.0008$ seconds

Question 4: Using the above approximations, calculate the baseband bandwidths for the PAM, PWM and PPM signals. Do these values reflect what you observe in the spectral plots? Note any discrepancies.

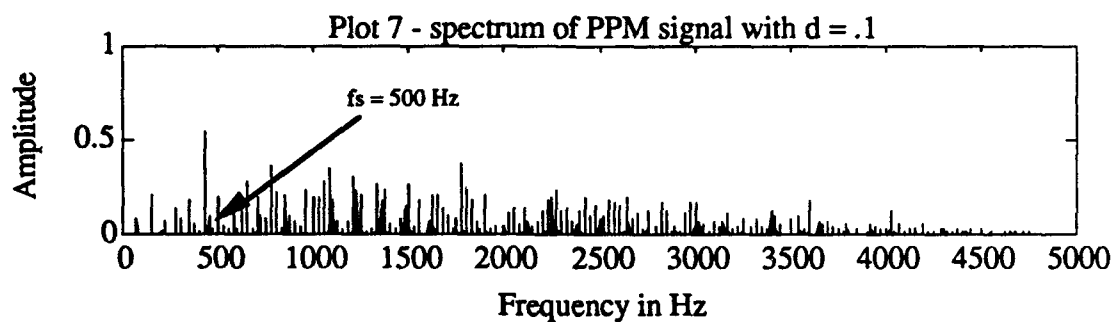
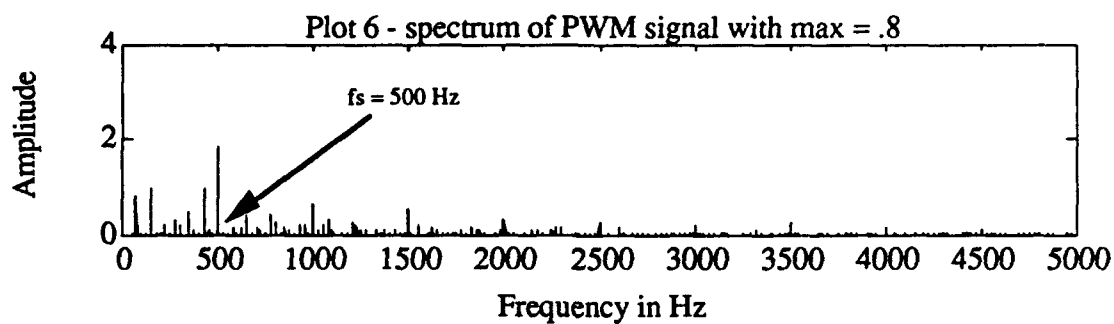
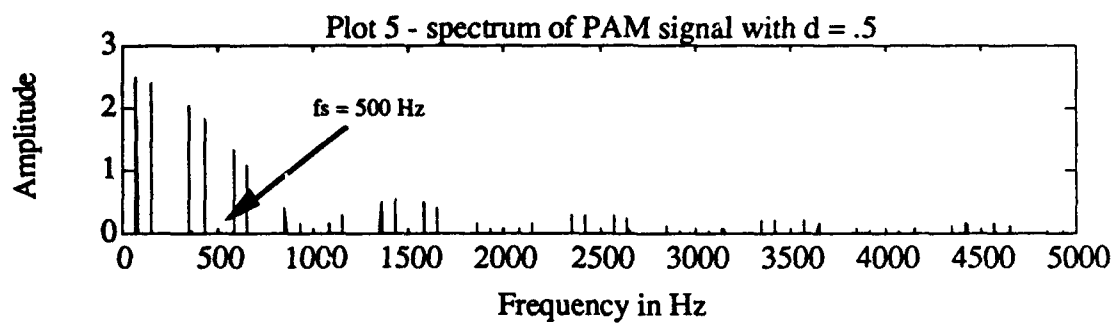
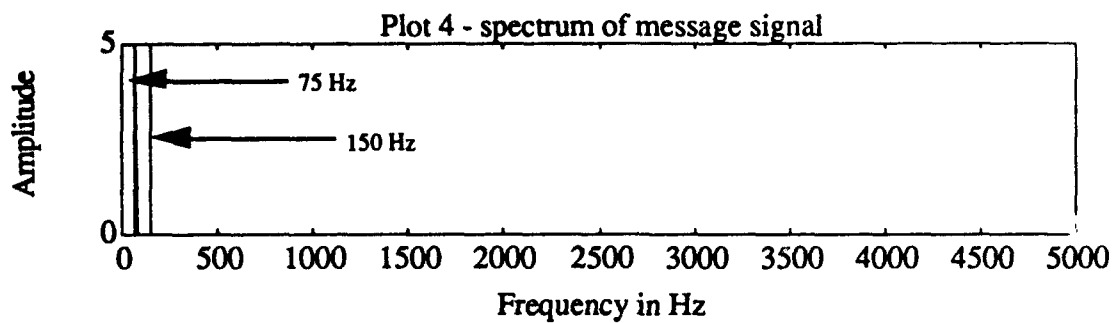
Answer: PAM bandwidth = $0.5/\tau \Rightarrow 0.5/0.001 \Rightarrow 500$ Hz
PWM and PPM = $0.5/\text{risetime} \Rightarrow 0.5/0.0001 = 5000$ Hz

The calculated baseband bandwidth of the PAM signal, 500 Hz, is adequate to capture the signal information. The PAM pulses occur at fixed, known intervals, and are of a fixed, known duration. The 500 Hz approximation is based solely on τ , the value of that duration.

PWM and PPM signals require a much higher baseband bandwidth because less information is known about their pulses. The higher frequencies are needed to convey the information regarding the exact locations or widths of the pulses. The approximation of 5000 Hz appears to capture most of the information required for the PWM and PPM signals.



NO PLOT HERE--JUST PRESS RETURN



lab3scr.m

%Computer-aided Lab 3 script for student use

%%

%Computer-aided Lab 3 Pulse Modulation (PAM, PWM, PPM)

%%

%Part 1--Observe the differences in the time domain for the

%three types of modulation

%A. Generating the signal

clg

clear

delta_t=.0001;

samprate=500;

t=0:delta_t:1;

s=5*(cos(2*pi*75*t)+sin(2*pi*150*t));

%max(s) %find the max and min values of s

%min(s)

%Plot 1

subplot(211), %plot the message signal

plot(t(101:500),s(101:500))

title('Plot 1 - message signal, PAM signal with d = .5')

xlabel('Time')

ylabel('Amplitude')

hold on

pause

%B. Modulating the signal

flatsig=flatop(s,delta_t,samprate,.5); %flatop sample the signal (PAM)

plot(t(101:500),flatsig(101:500),'b') %plot the pulse-amplitude modulated signal

hold off

pause

pwsig=pulswid(s,delta_t,samprate,.8); %pulse-width modulate the signal

%Plot 2

subplot(212),

```

plot(t(101:500),[s(101:500);pwsig(101:500)]) %plot the pulse-width modulated
title('Plot 2 - message signal, PWM signal with max = .8') %and message signal
xlabel('Time')
ylabel('Amplitude')

```

```

pause
clg

```

```

ppsig=pulspos(s,delta_t,samprate,.1); %pulse-position modulate the signal

```

```

%Plot 3

```

```

subplot(211),
plot(t(201:400),[s(201:400);ppsig(201:400)]) %plot message and pulse-position
title('Plot 3 - message signal, PPM signal with d = .1') %modulated signals
xlabel('Time')
ylabel('Amplitude')
grid

```

```

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

```

```

pause
clg

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Part 2--Observe the differences in the frequency domain for the
%three types of modulation
%A. Generating the spectra

```

```

[spec_s,HZ]=spectral(s,delta_t); %generate spectrum of the message signal

```

```

%Plot 4

```

```

subplot(211), %plot message spectrum
plot(HZ,spec_s)
title('Plot 4 - spectrum of message signal')
xlabel('Frequency in Hz')
ylabel('Amplitude')

```

```

clear s;spec_s;

```

```

specpam=spectral(flatsig,delta_t); %generate spectrum of the PAM signal

```

```

%Plot 5

```

```

subplot(212), %plot PAM spectrum
plot(HZ,specpam)

```

```

title('Plot 5 - spectrum of PAM signal with d = .5')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear flatsig;specpam;
pause
clg

specpw=spectral(pwsig,delta_t); %generate spectrum of the PWM signal

%Plot 6

subplot(211), %plot PWM spectrum
plot(Hz,specpw)
title('Plot 6 - spectrum of PWM signal with max = .8')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear pwsig;specpw;

specpp=spectral(ppsig,delta_t); %generate spectrum of the PPM signal

%Plot 7

subplot(212), %plot PPM spectrum
plot(Hz,specpp)
title('Plot 7 - spectrum of PPM signal with d = .1')
xlabel('Frequency in Hz')
ylabel('Amplitude')

```

EO 3513 Computer-aided Laboratory 4 Key
Analog-to-Digital Conversion and Digital Encoding

Question 1: Calculate the following values relating to the quantization characteristic for this system:

dynamic range
actual step size
actual resolution
percentage resolution
number of levels

Would you describe this quantizer as “mid-step,” or “mid-tread”?

Answer: dynamic range = $6 * \text{number of bits} \Rightarrow 6 * 5 \Rightarrow 30 \text{ dB}$
 actual step size = $2^{-n+1} \times \text{full-scale V} \Rightarrow \pm 2^{-5+1} \times 10 \Rightarrow 0.625 \text{ V}$
 actual resolution = $\pm 2^{-n} \times \text{full-scale V} \Rightarrow \pm 2^{-5} \times 10 \Rightarrow 0.3125 \text{ V}$
 percentage resolution = $\pm 2^{-n} \times 100\% \Rightarrow \pm 2^{-5} \times 100\% \Rightarrow 3.125\%$
 number of levels = $5 \text{ bits} \Rightarrow 2^5 \Rightarrow 32 \text{ levels}$

This quantizer is “mid-tread.”

Question 2: List the amplitude (“voltage”) of the quantized signal in each of the first 6 sampling periods.

Answer: 9.375 V
 8.125 V
 5.0 V
 0.0 V
 -4.375 V
 7.5 V

Question 3: From Plot 4, obtain the value of the signal-to-noise ratio for the quantized signal. Record this value.

Answer: 10.45 dB

Question 4: From the command window, obtain the first 30 values of "codedsig." Record these values. Is this bit pattern reflected on Plots 6, 9, and 12?

Answer:

1	1	1	1	0	1	1	1	0	0
1	0	1	1	1	1	0	0	0	0
0	1	0	0	1	0	0	1	0	0

Yes—the bit pattern is reflected on the plots.

Question 5: Describe the distinguishing characteristics of each encoding scheme:

**NRZL unipolar
RZL unipolar
manchester**

Answer: The NRZL unipolar encoded signal indicates a mark by remaining at some voltage level throughout the bit duration; it indicates a space by dropping to zero.

The RZL unipolar encoded signal indicates a mark by remaining at some voltage level for the first half of the bit duration, then dropping to zero for the last half; it indicates a space by remaining at zero for the bit duration.

The manchester encoded signal indicates a mark by remaining at some voltage level for the first half of the bit duration, then dropping to a second voltage level for the last half of the bit duration; it indicates a space by remaining at the lower voltage level for the first half of the bit duration, and rising to the higher level for the second half of the bit duration.

Question 6: Calculate the approximate baseband bandwidth for the PCM signals:

**NRZL unipolar coded signal
RZL and manchester coded signals**

Do these values reflect what you observe in the spectral plots?

Answer: For NRZL signal: $B = 0.5/\tau \Rightarrow 0.5/0.001 \text{ seconds} \Rightarrow 500 \text{ Hz}$

For RZL and manchester signals: $B = 0.5/\tau \Rightarrow 1/0.0005 \text{ seconds} \Rightarrow 1000 \text{ Hz}$

The spectral plots support the above approximations.

Question 7: From Plot 16, obtain the value of the signal-to-noise ratio for the quantized signal. Record this value.

Answer: 10.97 dB

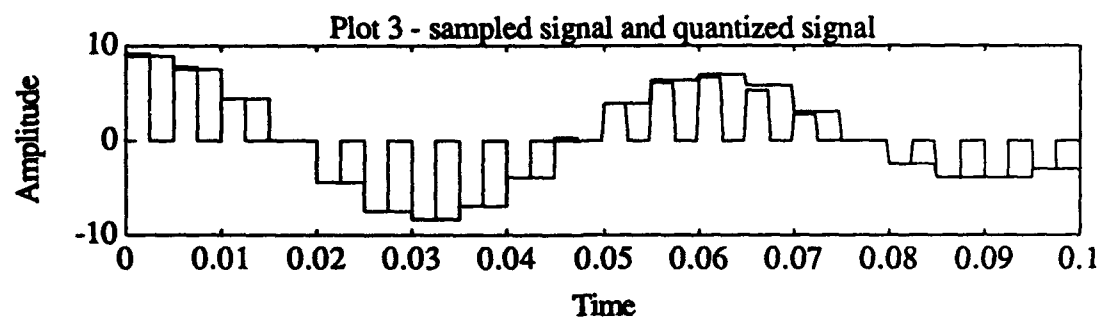
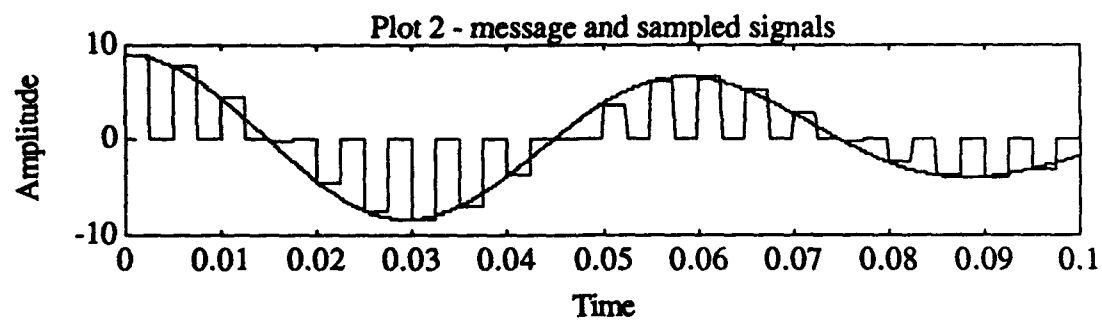
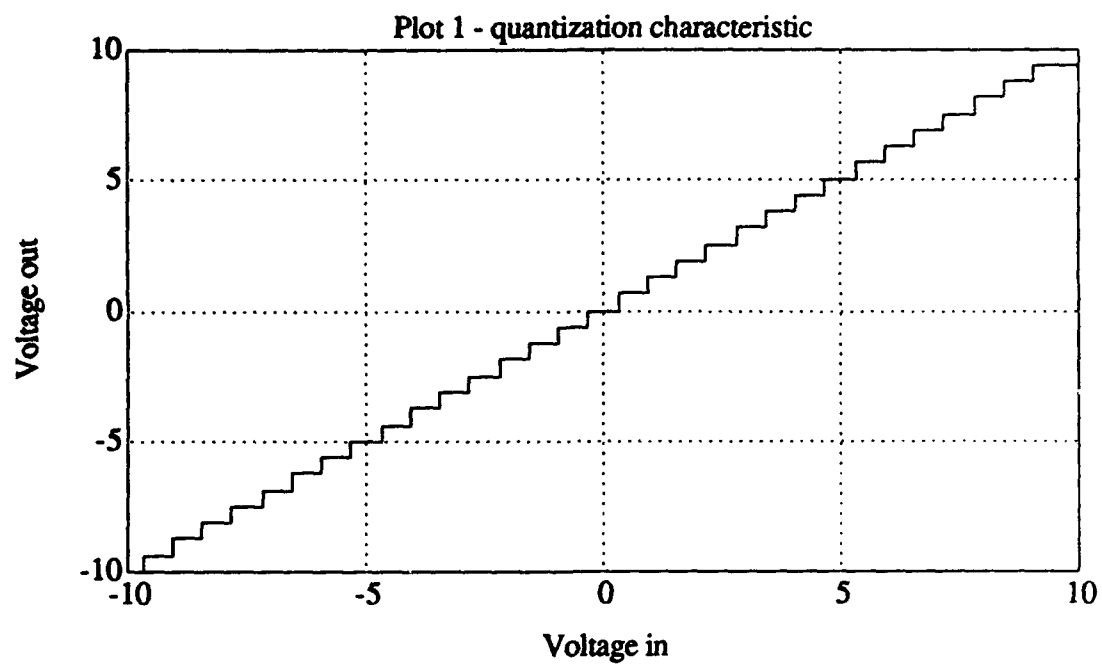
Question 8: What is the effect of compression on the signal?

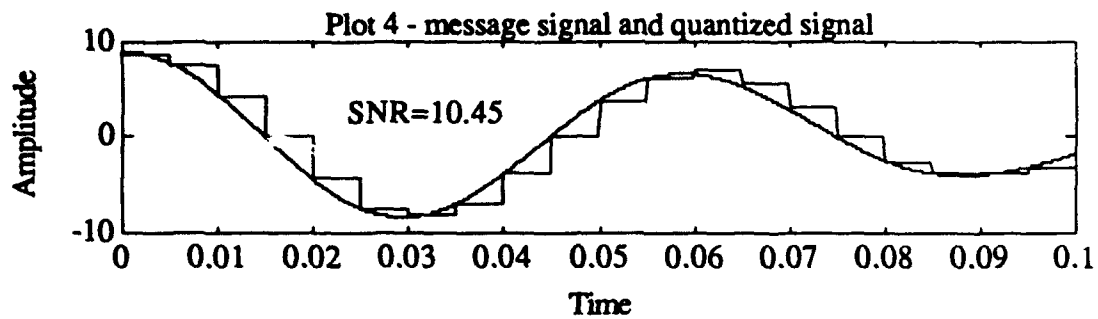
Answer: The compression function increases the lower-amplitude signal values in a manner that minimizes the extreme differences in the signal.

Question 9: From Plot 21, obtain the value of the signal-to-noise ratio for the companded signal. Record this value and compare it to the ratio obtained in Question 7. Did the companding process reduce the amount of quantization noise?

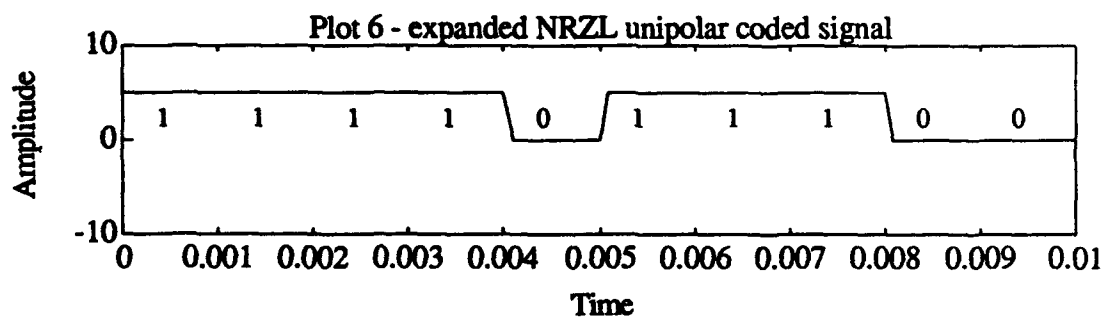
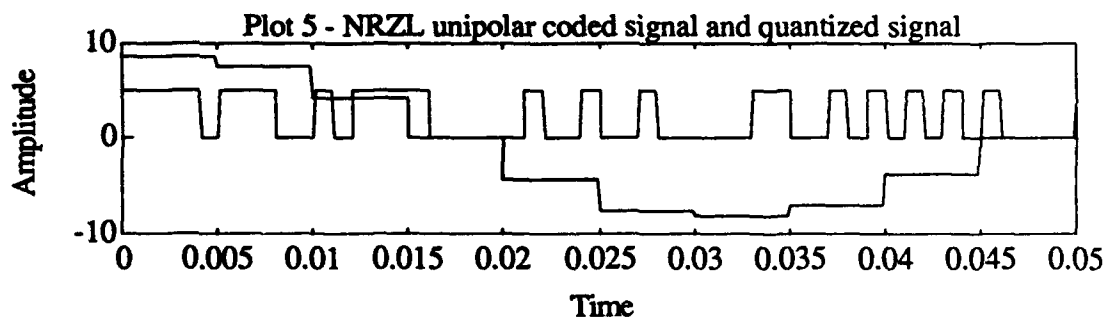
Answer: 13.03 dB

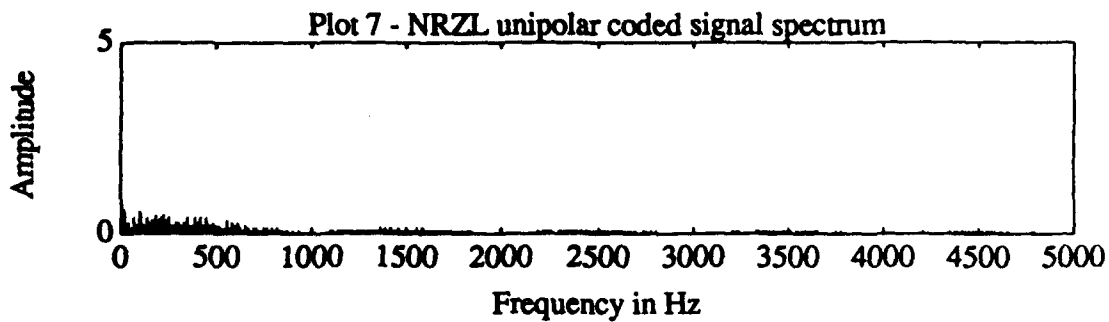
Yes—the signal to noise ratio increased due to use of the compression and expansion functions.



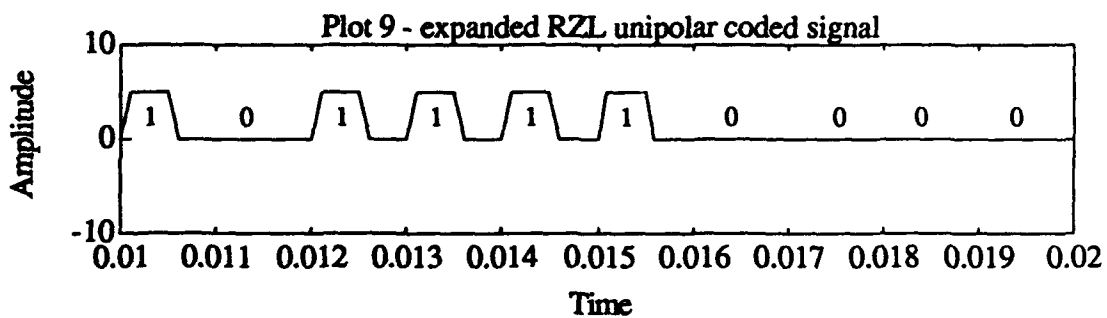
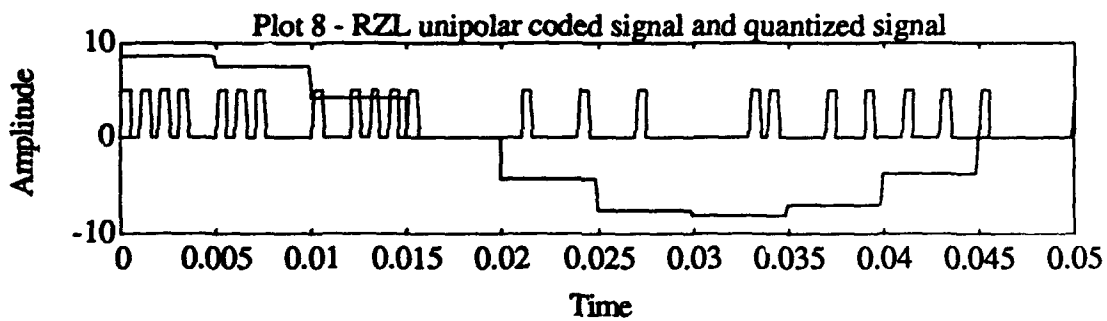


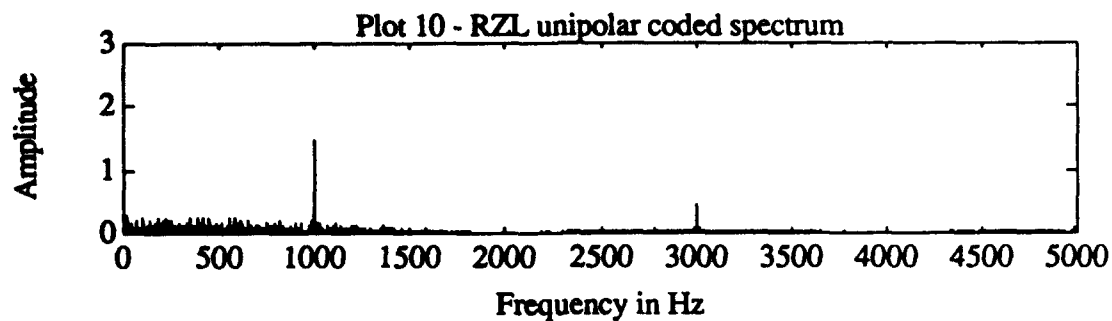
NO PLOT HERE--JUST PRESS RETURN



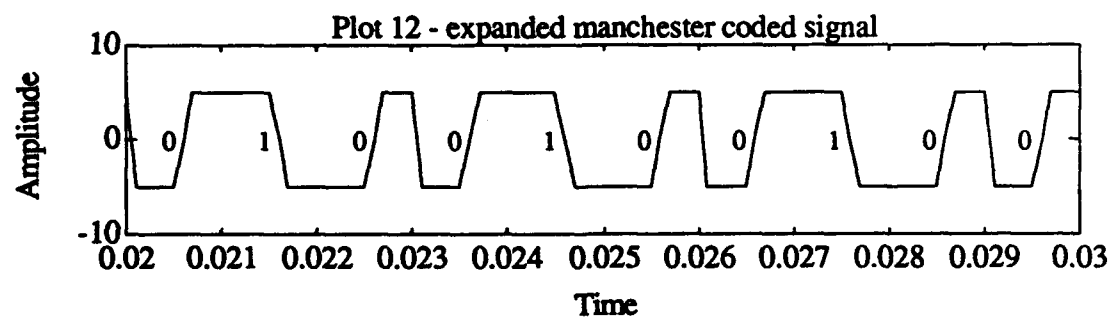
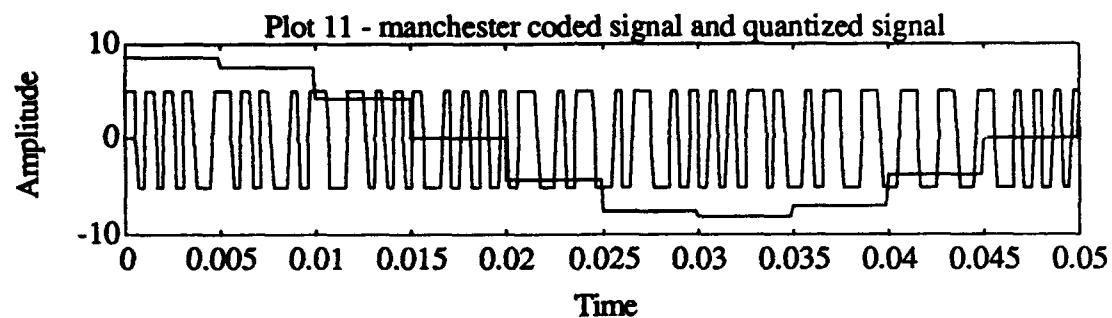


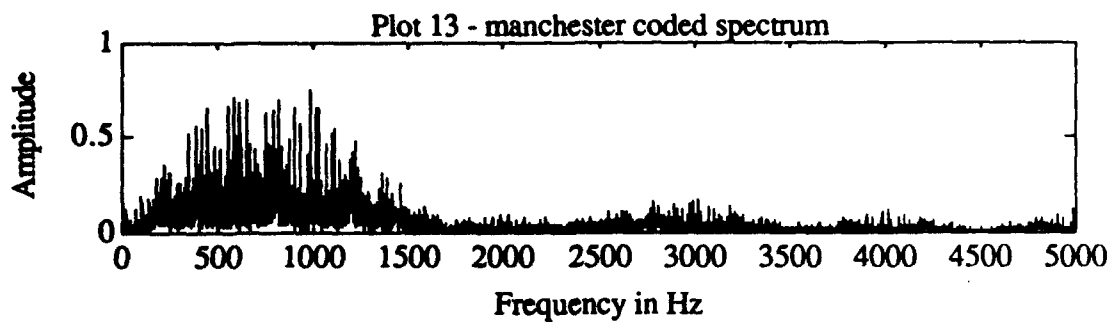
NO PLOT HERE--JUST PRESS RETURN



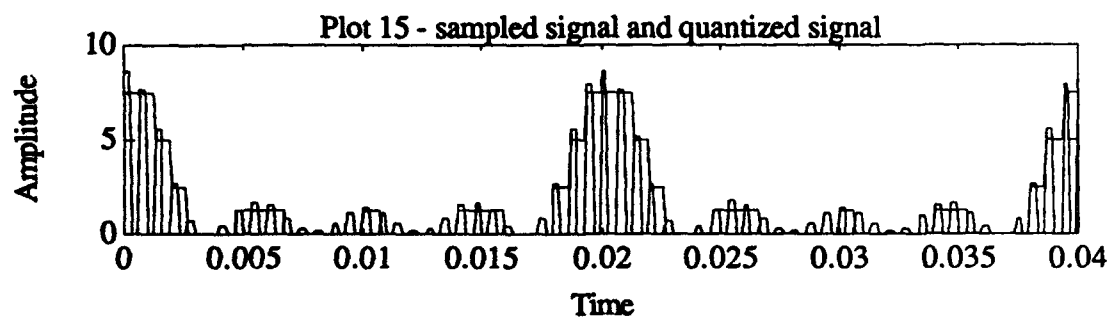
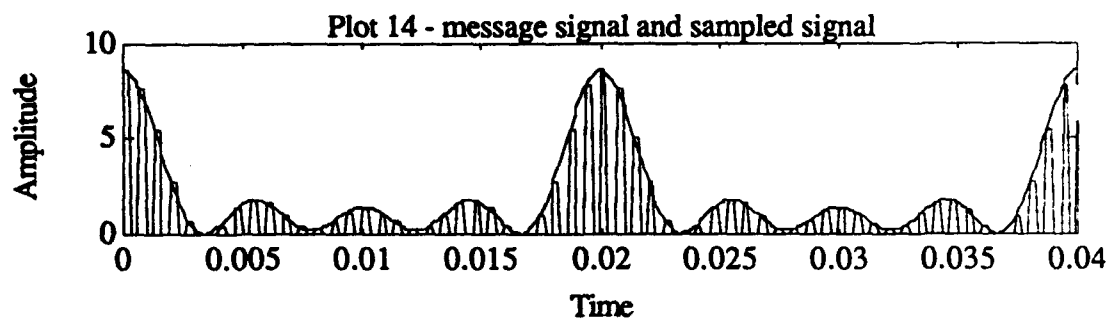


NO PLOT HERE--JUST PRESS RETURN

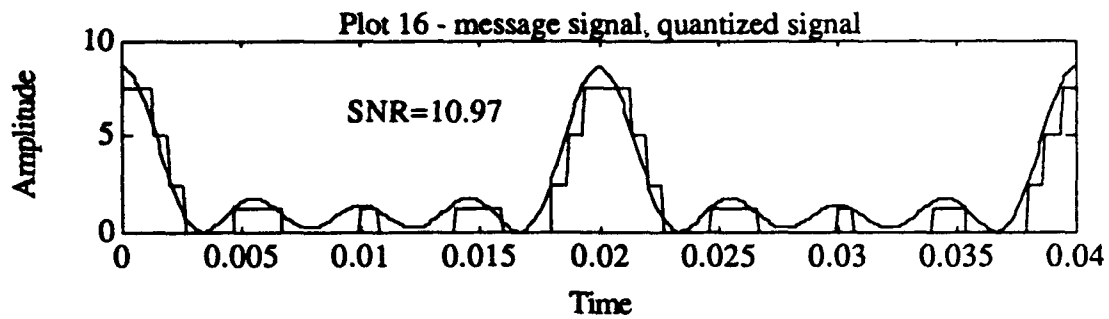




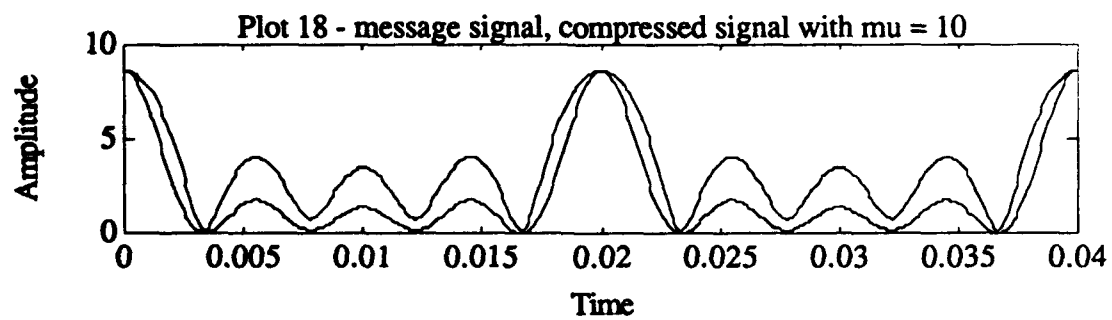
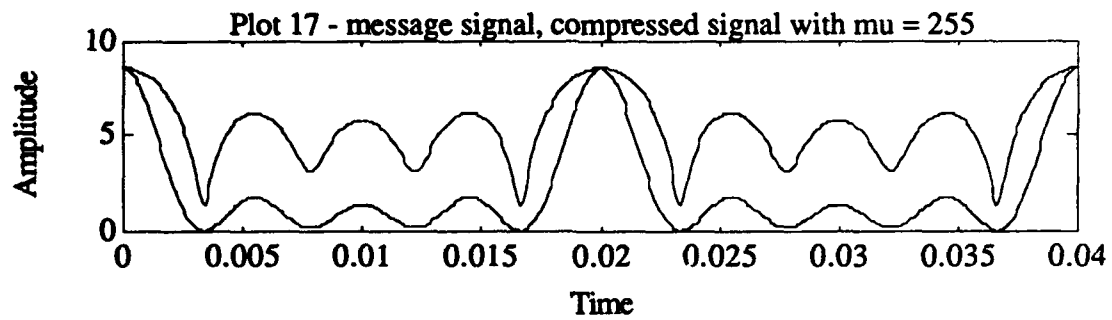
NO PLOT HERE--JUST PRESS RETURN

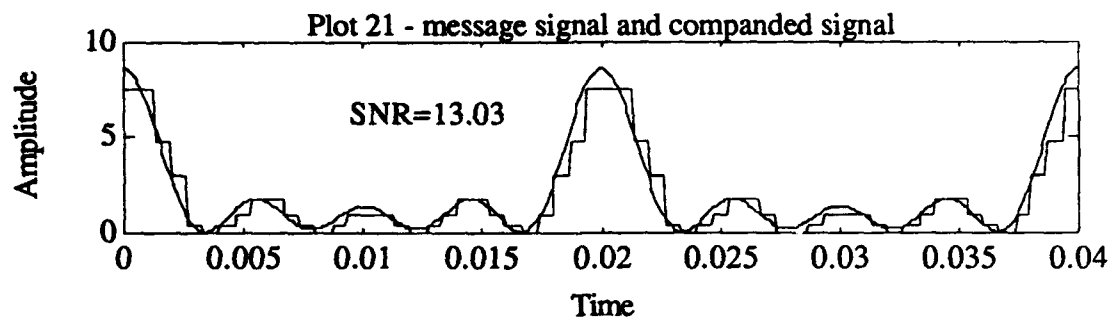
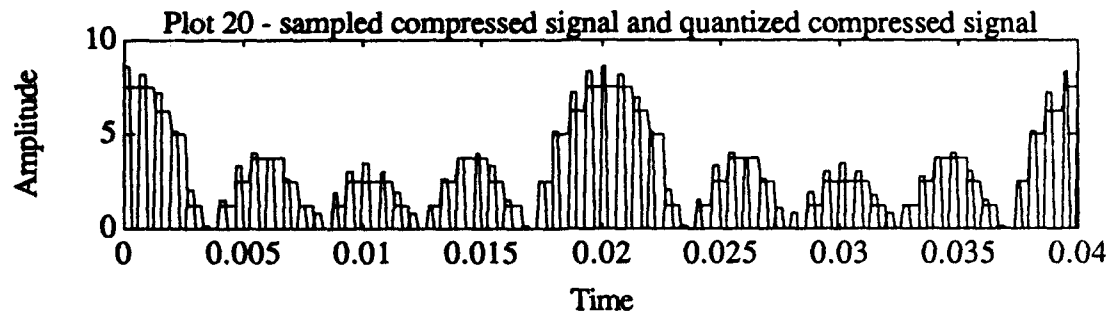
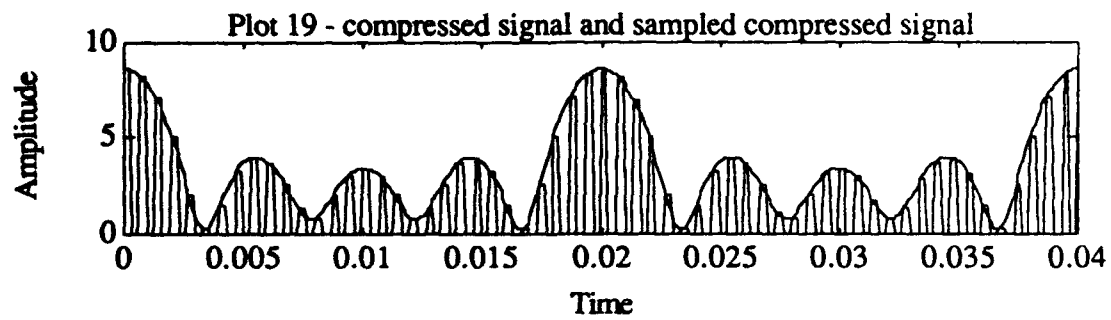


Computer-aided Laboratory 4 Key—page 8



NO PLOT HERE--JUST PRESS RETURN





NO PLOT HERE

lab4scr.m

%Computer-aided Lab 4 script for student use

%%
%Computer-aided Lab 4 Analog-to-Digital Conversion and
% Digital Encoding

%%

%Part 1—Observe the quantization process

%A. Generating the message and sampled signals

clg
clear

delta_t=.0001; %set signal and sampling variables
d=.5;
samprate=200;

t=0:delta_t:1;
s1=5*cos(2*pi*15*t)+4*cos(2*pi*19*t); %signal and sampling frequencies
%are intentionally low
flatsig=flatop(s1,delta_t,samprate,d);

%B. Evaluate the characteristics of the converter

[quanch_x,quanch_y,quansig,bin_nms]=quantize(2,5,flatsig,samprate,delta_t);

%Plot 1

stairs(quanch_x,quanch_y)
grid
title('Plot 1 - quantization characteristic')
xlabel('Voltage in')
ylabel('Voltage out')

pause
clg

%C. Compare the sampled and quantized signals

%Plot 2

subplot(211), %plot the message and sampled signals
plot(t(1:1000),[s1(1:1000);flatsig(1:1000)])
title('Plot 2 - message and sampled signals')
xlabel('Time')
ylabel('Amplitude')

%Plot 3

Computer-aided Laboratory 4 Key—page 11

```

subplot(212),
plot(t(1:1000),[flatsig(1:1000);quansig(1:1000)])
title('Plot 3 - sampled signal and quantized signal')
xlabel('Time')
ylabel('Amplitude')

pause
clg

quanch_y(1:(length(quanch_y)-1))
some_bin_nums=bin_nums(1:6)

pause

%L. Measure the quantization noise

sig_to_nois=snr(s1,quansig);
str=num2str(sig_to_nois);

%Plot 4

subplot(211),
plot(t(1:1000),[s1(1:1000);quansig(1:1000)])
title('Plot 4 - message signal and quantized signal')
xlabel('Time')
ylabel('Amplitude')
text(.3,.8,['SNR=' str],'sc')

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

pause
clg

%%%%%%%%%%%%%%
%Part 2--Observe the process of PCM encoding
%A. Generating a binary-encoded signal

codedsig=encode(bin_nums,2,5); %binary-encode the signal

%B. Generating a non-return-to-zero level (NRZL) coded signal

bitrate=samprate*5;

nrzlunisig=nrzluni(codedsig,delta_t,bitrate);
nrzlunisig=nrzlunisig*5; %increase signal level for plotting

```

```

pcm_axis=[0 .05 -10 10];
pcm_axis1=[0 .01 -10 10];
pcm_axis2=[.01 .02 -10 10];
pcm_axis3=[.02 .03 -10 10];

axis(pcm_axis);

%Plot 5

subplot(211),
plot(t(1:600),[quansig(1:600);nrzlunisig(1:600)])
title('Plot 5 - NRZL unipolar coded signal and quantized signal')
xlabel('Time')
ylabel('Amplitude')

axis(pcm_axis1);

%Plot 6

subplot(212),
plot(t(1:600),nrzlunisig(1:600))
title('Plot 6 - expanded NRZL unipolar coded signal')
xlabel('Time')
ylabel('Amplitude')

axis;

pause
clg

[specnrzl,Hz]=spectral(nrzlunisig,delta_t);
clear nrzlunisig;

%Plot 7

subplot(211),
plot(Hz,specnrzl)
title('Plot 7 - NRZL unipolar coded signal spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

pause
clg

clear specnrzl

```


%C. Generating a return-to-zero level (RZL) unipolar coded signal

```
rzunisig=rzuni(codedsig,delta_t,bitrate);
rzunisig=rzunisig*5; %increase signal level for plotting

axis(pcm_axis);

%Plot 8

subplot(211),
plot(t(1:600),[quansig(1:600);rzunisig(1:600)])
title('Plot 8 - RZL unipolar coded signal and quantized signal')
xlabel('Time')
ylabel('Amplitude')

axis(pcm_axis2);

%Plot 9

subplot(212),
plot(t(1:600),rzunisig(1:600))
title('Plot 9 - expanded RZL unipolar coded signal')
xlabel('Time')
ylabel('Amplitude')

axis;

pause
clg

specrz=spectral(rzunisig,delta_t);
clear rzunisig

%Plot 10

subplot(211),
plot(Hz,specrz)
title('Plot 10 - RZL unipolar coded spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

pause
clg
```

```

clear specrz

%D. Generating a manchester coded signal

manchsig=manchest(codedsig,delta_t,bitrate);
manchsig=manchsig*5; %increase signal level for plotting

axis(pcm_axis);

%Plot 11

subplot(211),
plot(t(1:600),[quansig(1:600);manchsig(1:600)])
title('Plot 11 - manchester coded signal and quantized signal')
xlabel('Time')
ylabel('Amplitude')

axis(pcm_axis3);

%Plot 12

subplot(212),
plot(t(1:600),manchsig(1:600))
title('Plot 12 - expanded manchester coded signal')
xlabel('Time')
ylabel('Amplitude')

axis;

pause
clg

specman=spectral(manchsig,delta_t);
clear manchsig
some_codedsig=codedsig(1:30) %print out values of first 6 words
pause

%Plot 13

subplot(211),
plot(Hz,specman)
title('Plot 13 - manchester coded spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

```

```

pause

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Part 3—Observe the effects of companding on the quantization
%   process
%A. Generating a signal

clear
clf

samprate=1500;
d=.5;
delta_t=.0001;

t=0:delta_t:0.1;

s=2+2.1*cos(2*pi*50*t)+1.7*cos(4*pi*50*t)+1.5*cos(6*pi*50*t);
s=s+1.3*cos(8*pi*50*t);

comp_axis=[0 0.04 0 10]; %set plotting boundaries
axis(comp_axis);

mu1=255; %high value of mu
mu2=10;  %low value of mu

%Plot 14

subplot(211),
plot(t(1:1000),s(1:1000))
title('Plot 14 - message signal and sampled signal')
xlabel('Time')
ylabel('Amplitude')
hold on

%B. Sampling and quantizing the signal

flatsig1=flattop(s,delta_t,samprate,d);

plot(t(1:1000),flatsig1(1:1000),'g')
hold off

axis(comp_axis);

%Plot 15

subplot(212),
plot(t(1:1000),flatsig1(1:1000))
title('Plot 15 - sampled signal and quantized signal')

```

```

xlabel('Time')
ylabel('Amplitude')
hold on

[quanch_x,quanch_y,quansig1]=quantuni(2,3,flatsig1,samprate,delta_t);

plot(t(1:1000),quansig1(1:1000),'b')
hold off

axis(comp_axis);

pause
clg

snr_quant=snr(s,quansig1); %find the quantization noise
str=num2str(snr_quant);

%Plot 16

subplot(211),
plot(t(1:1000),[s(1:1000);quansig1(1:1000)])
title('Plot 16 - message signal, quantized signal')
xlabel('Time')
ylabel('Amplitude')
text(.3,.8,['SNR=' str],'sc')

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

pause
clg

%C. Compare compression characteristics for values of mu

press1_s=compress(s,mu1,max(s)); %use signal maximum for compression function

%Plot 17

subplot(211),
plot(t(1:1000),[s(1:1000);press1_s(1:1000)])
title('Plot 17 - message signal, compressed signal with mu = 255')
xlabel('Time')
ylabel('Amplitude')

press2_s=compress(s,mu2,max(s));
exp_sig=expand(press2_s,mu2,max(press2_s));

%Plot 18

```

```

subplot(212),
plot(t(1:1000),[s(1:1000);press2_s(1:1000)])
title('Plot 18 - message signal, compressed signal with mu = 10')
xlabel('Time')
ylabel('Amplitude')

pause
clg

%D. Sampling and quantizing the compressed signal
      %use 8 levels to illustrate

flatsig2=flattop(press2_s,delta_t,samprate,d);

%Plot 19

subplot(211),
plot(t(1:1000),[press2_s(1:1000);flatsig2(1:1000)])
title('Plot 19 - compressed signal and sampled compressed signal')
xlabel('Time')
ylabel('Amplitude')

[quanch_x,quanch_y,quansig2]=quantuni(2,3,flatsig2,samprate,delta_t);

%Plot 20

subplot(212),
plot(t(1:1000),[flatsig2(1:1000);quansig2(1:1000)])
title('Plot 20 - sampled compressed signal and quantized compressed signal')
xlabel('Time')
ylabel('Amplitude')

pause
clg

%E. Expand the compressed quantized signal

exp_comp=expand(quansig2,mu2,max(quansig2)); %use maximum signal value
snr_comp=snr(s,exp_comp);
str=num2str(snr_comp);

%Plot 21

subplot(211),
plot(t(1:1000),[s(1:1000);exp_comp(1:1000)])
title('Plot 21 - message signal and companded signal')
xlabel('Time')

```

```
ylabel('Amplitude')  
text(3,.8,['SNR=' str],'sc')
```

```
subplot(212),  
title('NO PLOT HERE')
```

```
axis;
```

**This page is intentionally
left blank.**

EO 3513 Computer-aided Laboratory 5 Key Amplitude Modulation Double Sideband (AM DSB)

Question 1: Calculate the following values for the single-tone message signal:

peak power
average power
baseband bandwidth

Answer: peak power $P_P = \frac{A_P^2}{2} \Rightarrow 10^2 / 2 \Rightarrow 50$

average power $P = A_0^2 + \frac{1}{2} \sum_{N=1}^{\infty} (A_N^2 + B_N^2) \Rightarrow 10^2 / 2 \Rightarrow 50$

baseband bandwidth 150 Hz

Question 2: Predict the following values for the single-tone AM DSB signal:

peak power
average power
bandwidth

Answer: peak power $P_P = \frac{A_P^2}{2} \Rightarrow 10^2 / 2 \Rightarrow 50$

average power $P = \frac{A^2}{4} \Rightarrow 10^2 / 4 \Rightarrow 25$

bandwidth 300 Hz

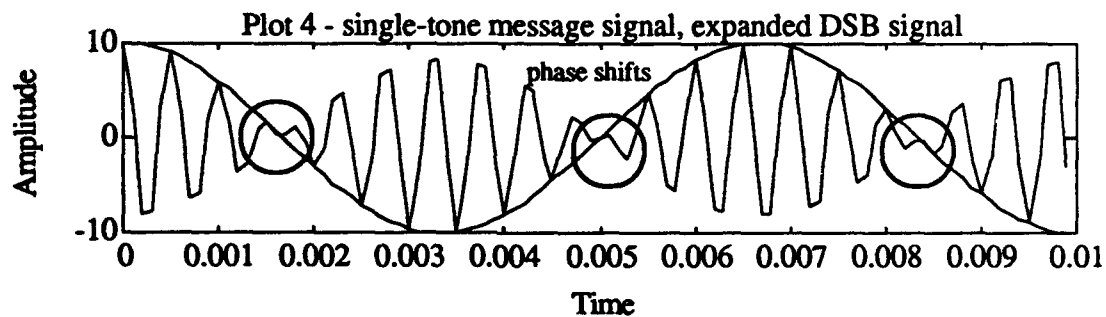
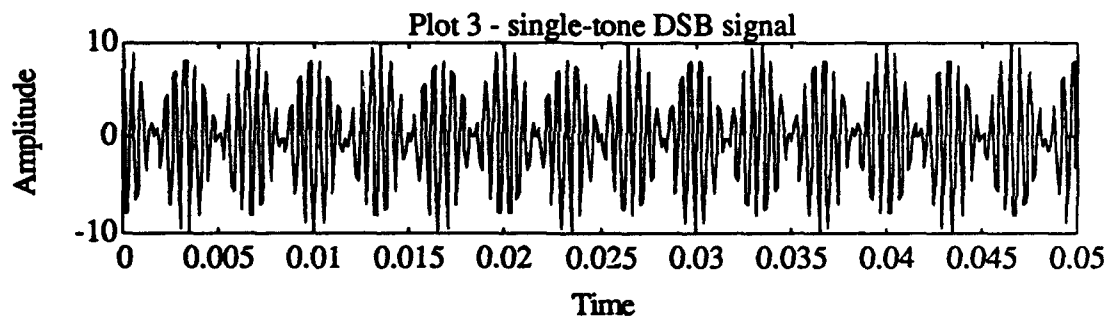
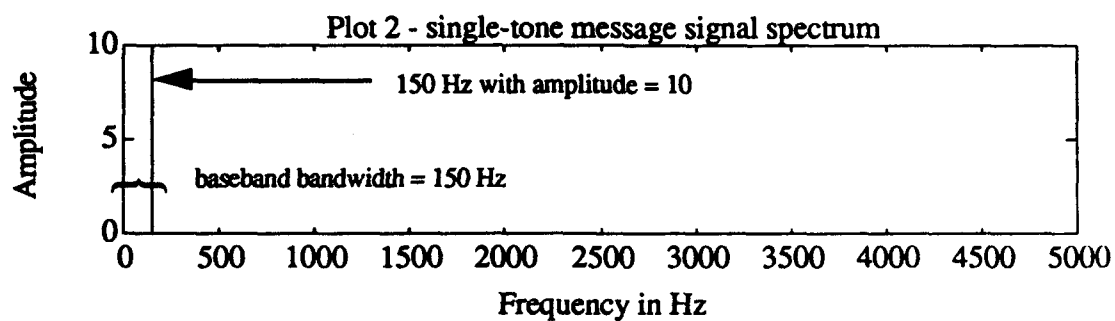
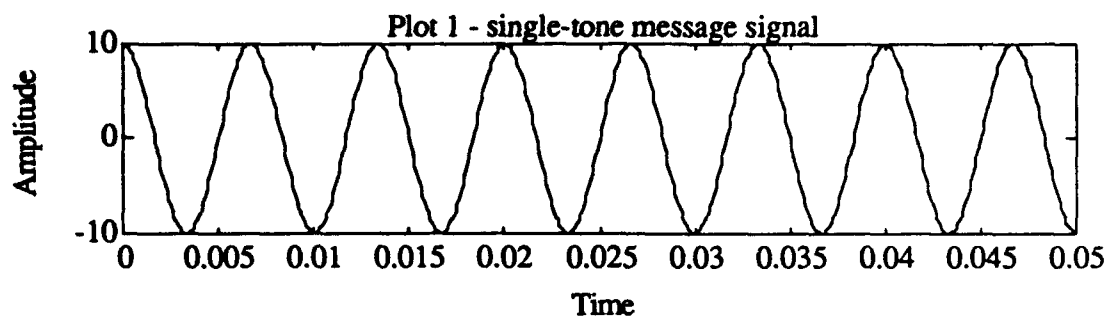
Question 3: From Plot 6, obtain the values representing peak and average power for the signal-tone signal, and record them.

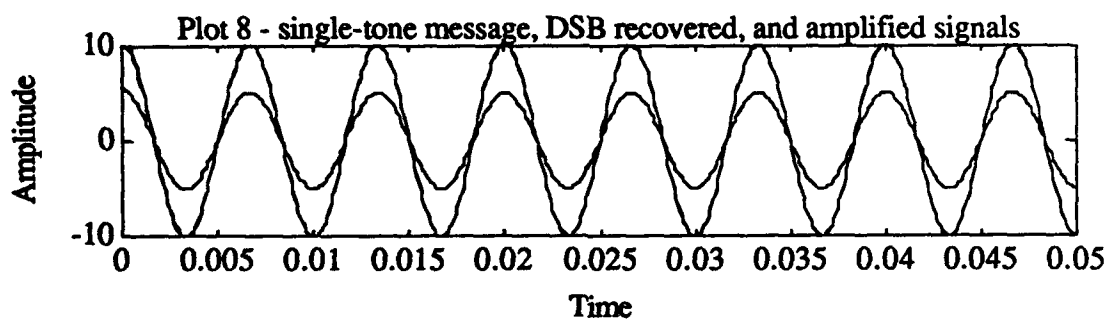
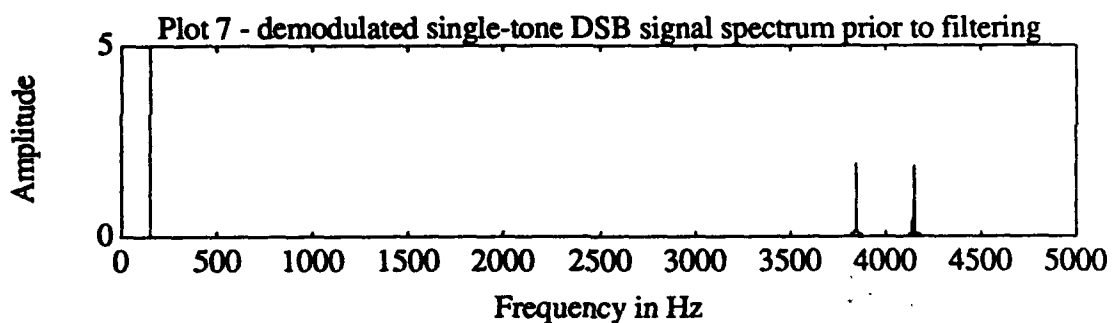
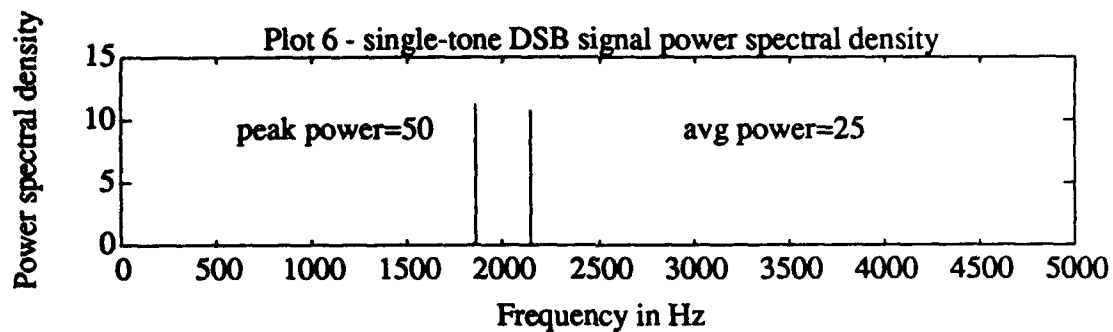
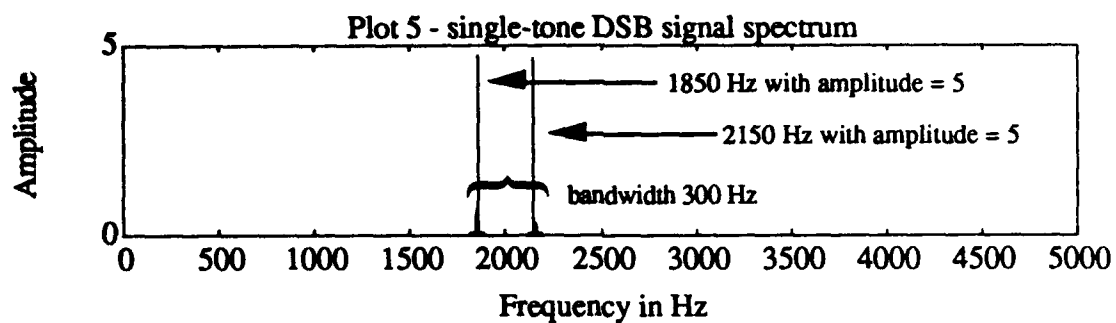
Do your calculations for bandwidth and power agree with the computer-generated values and spectrum?

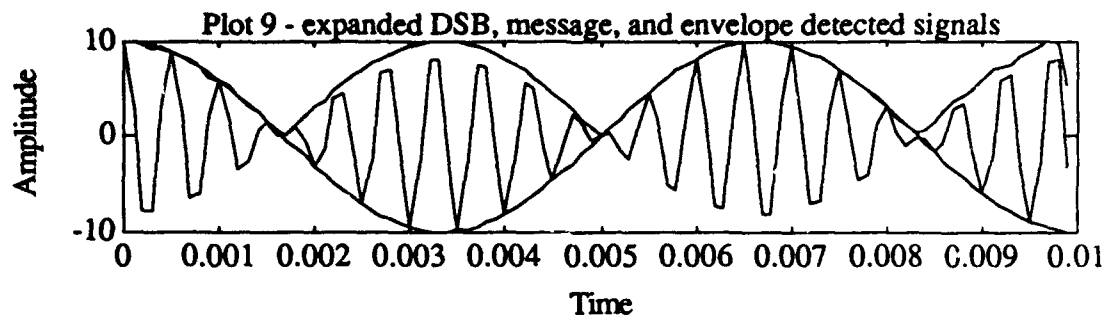
Answer: peak power = 50
average power = 25
Yes—calculations agree.

Question 4: Why is coherent detection (detection using the carrier) necessary for an AM DSB signal?

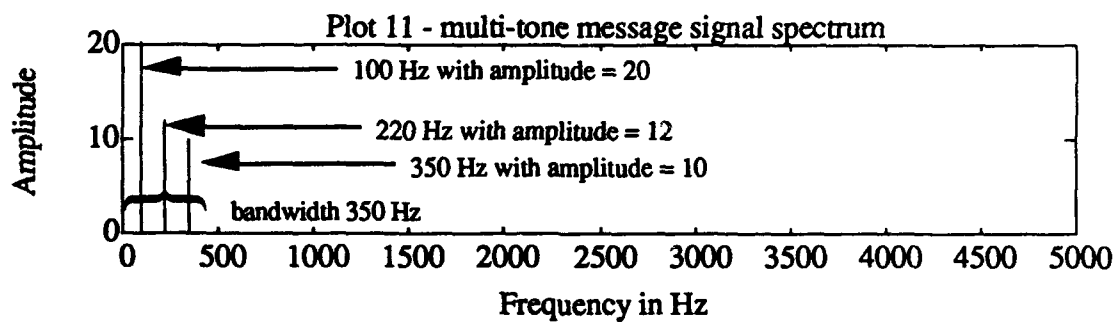
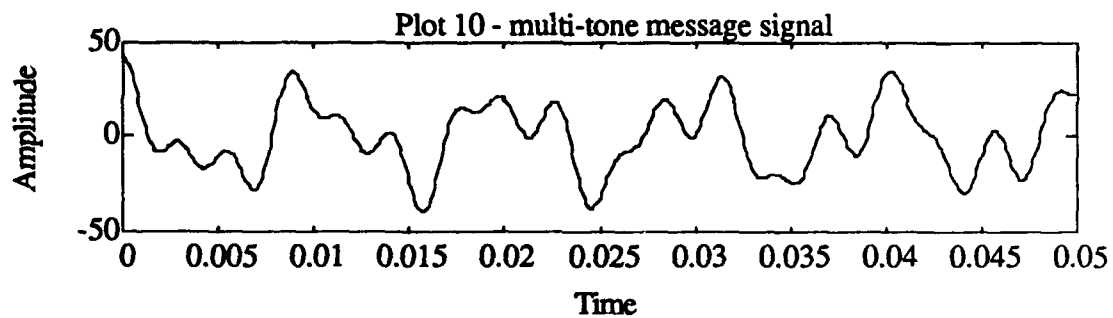
Answer: Envelope detection of the AM DSB signal would not detect phase shifts, which indicate that the message signal has changed from positive to negative values, or from negative to positive values.

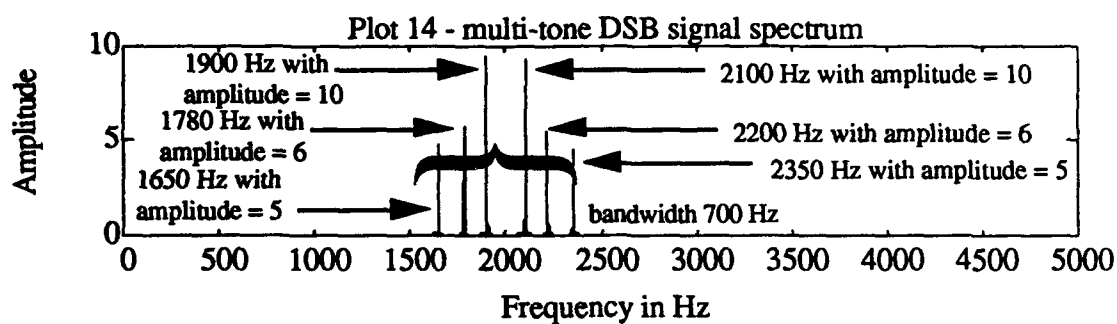
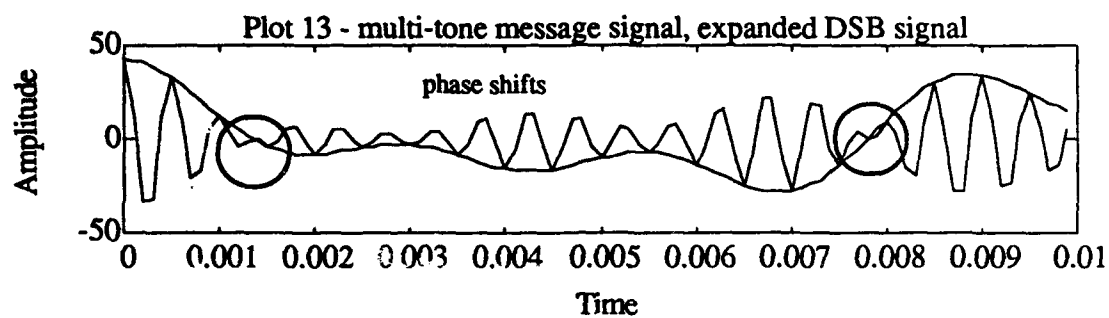
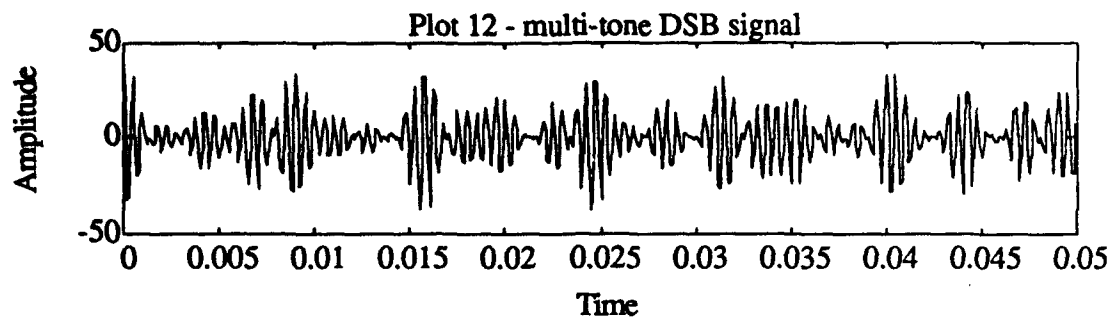




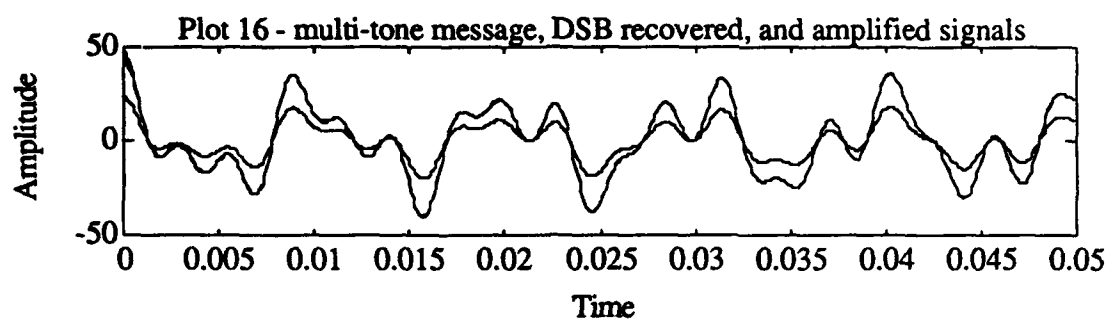
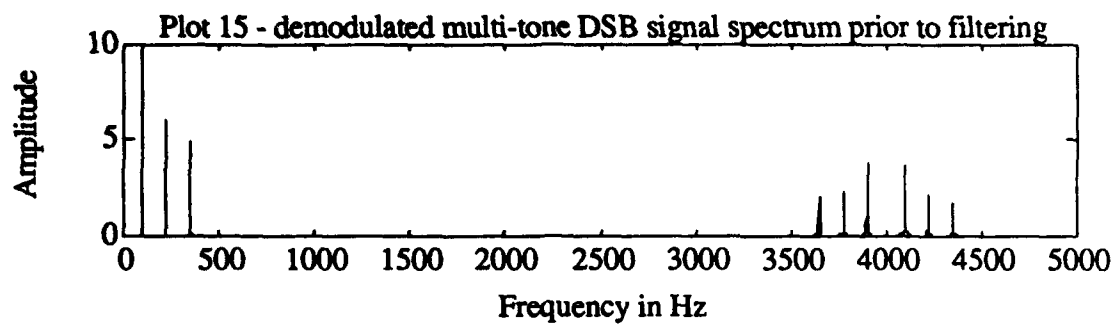


NO PLOT HERE--JUST PRESS RETURN





NO PLOT HERE--JUST PRESS RETURN



lab5scr.m

%Computer-aided Lab 5 script for student use

%%
%Computer-aided Lab 5 Amplitude Modulation AM DSB
%%
%Part 1—Observe the AM DSB modulation process with single-tone
% input
%A. Generating the signal and spectrum

clear
clg

delta_t=.0001; %set signal and sampling variables
t=0:delta_t:1;

s=10*cos(2*pi*150*t); %single-tone signal variable

fc=2000; %modulating frequency for the carrier signal
cutoff=160; %ideal lowpass filter cutoff frequency for single-tone

%Plot 1

subplot(211), %plot the signal
plot(t(1:500),s(1:500))
title('Plot 1 - single-tone message signal')
xlabel('Time')
ylabel('Amplitude')

[specs,HZ]=spectral(s,delta_t); %generate the spectrum

%Plot 2

subplot(212), %plot the spectrum
plot(HZ,specs)
title('Plot 2 - single-tone message signal spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

%B. Observe the single-tone double sideband (DSB) modulated
% signal

moddsbs=cos(2*pi*fc*t).*s; %modulate the signal by multiplying by a cosine

%Plot 3

```
subplot(211), %plot the modulated signal
plot(t(1:500),moddsbs(1:500))
title('Plot 3 - single-tone DSB signal')
xlabel('Time')
ylabel('Amplitude')
```

%Plot 4

```
subplot(212), %plot detailed view to show phase shifts
plot(t(1:100),[s(1:100);moddsbs(1:100)])
title('Plot 4 - single-tone message signal, expanded DSB signal')
xlabel('Time')
ylabel('Amplitude')
```

```
pause
clg
```

%C. Verify the power and bandwidth of the DSB signal

```
modspecdsbs=spectral(moddsbs,delta_t); %generate the modulated spectrum
```

%Plot 5

```
subplot(211), %plot the modulated spectrum
plot(Hz,modspecdsbs)
title('Plot 5 - single-tone DSB signal spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')
```

```
clear modspecdsbs;
```

```
dsb_pk_pwr_sngl=((max(moddsbs))^2)/2; %find the peak power
str1=num2str(dsb_pk_pwr_sngl);
psddb=psd(moddsbs,delta_t); %generate the power spectral density

dsb_avg_pwr_sngl=sum(psddb); %find average power by summing the power
                           %spectral density values
str2=num2str(dsb_avg_pwr_sngl);
```

%Plot 6

```
subplot(212), %plot the power spectral density for the modulated signal
plot(Hz,psddb)
title('Plot 6 - single-tone DSB signal power spectral density')
xlabel('Frequency in Hz')
ylabel('Power spectral density')
```



```

text(2,.3,['peak power=' str1'],'sc')
text(6,.3,['avg power=' str2'],'sc')

clear psddsbs;

pause
clg

%D. Observe the recovery of the DSB signal

demoddsbs=cos(2*pi*fc*t).*moddsbs; %first step in recovering the signal--
                                     %multiply by the carrier

[recspecdsbs,HZ,dsbfft]=spectral(demoddsbs,delta_t); %generate the spectrum
                                                         %the recovered signal

%Plot 7

subplot(211),
plot(HZ,recspecdsbs)
title('Plot 7 - demodulated single-tone DSB signal spectrum prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear demoddsbs;clear recspecdsbs;
moddsbs=moddsbs(1:100);

recdsbs=recoverm(dsbfft,'ideallow',HZ,cutoff); %recover and filter
clear dsbfft;
bigrecdsbs=recdsbs*2; %amplify signal

%Plot 8

subplot(212), %plot the recovered signal on top of the message signal
plot(t(1:500),[s(1:500);recdsbs(1:500)])
title('Plot 8 - single-tone message, DSB recovered, and amplified signals')
xlabel('Time')
ylabel('Amplitude')
hold on
pause
plot(t(1:500),bigrecdsbs(1:500),'b')
hold off

pause
clg

detdsbs=cplxenv(moddsbs);

%Plot 9

```

```

subplot(211), %plot the recovered signal on top of the message signal
plot(t(1:100),[moddsbs(1:100);s(1:100)])
title('Plot 9 - expanded DSB, message, and envelope detected signals')
xlabel('Time')
ylabel('Amplitude')
hold on
pause
plot(t(1:100),detdsbs(1:100), 'b')
hold off

```

```

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

```

```

pause

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Part 2--Observe the AM DSB modulation process with
%    multi-tone input
%A. Generating the signal and spectrum

```

```

clear
clg

```

```

delta_t=.0001; %set signal and sampling variables
t=0:delta_t:1;

```

```

    %multi-tone signal variable
s=10*cos(2*pi*350*t)+12*cos(2*pi*220*t)+20*cos(2*pi*100*t);

```

```

fc=2000; %modulating frequency for the carrier signal
cutoff=360; %ideal lowpass filter cutoff frequency for multi-tone

```

```

%Plot 10

```

```

subplot(211), %plot the signal
plot(t(1:500),s(1:500))
title('Plot 10 - multi-tone message signal')
xlabel('Time')
ylabel('Amplitude')

```

```

[specs,HZ]=spectral(s,delta_t); %generate the spectrum

```

```

%Plot 11

```

```

subplot(212), %plot the spectrum
plot(HZ,specs)
title('Plot 11 - multi-tone message signal spectrum')
xlabel('Frequency in Hz')

```

```

ylabel('Amplitude')

pause
clg

%B. Observe the multi-tone double sideband (DSB) modulated
% signal and spectrum

moddsbs=cos(2*pi*fc*t).*s; %modulate the signal by multiplying by a cosine

%Plot 12

subplot(211), %plot the modulated signal
plot(t(1:500),moddsbs(1:500))
title('Plot 12 - multi-tone DSB signal')
xlabel('Time')
ylabel('Amplitude')

%Plot 13

subplot(212), %plot detailed view to show phase shifts
plot(t(1:100),[s(1:100);moddsbs(1:100)])
title('Plot 13 - multi-tone message signal, expanded DSB signal')
xlabel('Time')
ylabel('Amplitude')

pause
clg

modspecdsbs=spectral(moddsbs,delta_t); %generate the modulated spectrum

%Plot 14

subplot(211), %plot the modulated spectrum
plot(Hz,modspecdsbs)
title('Plot 14 - multi-tone DSB signal spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

pause
clg

%C. Observe the recovery of the DSB signal

demoddsbs=cos(2*pi*fc*t).*moddsbs; %first step in recovering the signal--

```

```

                                %multiply by the carrier

[recspecdsbs,Hz,dsbfft]=spectral(demoddsbs,delta_t); %generate the spectrum
                                %the recovered signal

%Plot 15

subplot(211),
plot(Hz,recspecdsbs)
title('Plot 15 - demodulated multi-tone DSB signal spectrum prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear recspecdsbs;clear specs;clear modspecdsbs;clear demoddsbs;
clear moddsbs;

recdsb = verrm(dsbfft,'ideallow',Hz,cutoff); %recover and filter

bigrecdsbs=recdsbs*2; %amplify signal

%Plot 16

subplot(212), %plot the recovered signal on top of the message signal
plot(t(1:500),[s(1:500);recdsbs(1:500)])
title('Plot 16 - multi-tone message, DSB recovered, and amplified signals')
xlabel('Time')
ylabel('Amplitude')
hold on
pause
plot(t(1:500),bigrecdsbs(1:500),'b')
hold off

```

**This page is intentionally
left blank.**

EO 3513 Computer-aided Laboratory 6 Key
Amplitude Modulation Single Sideband (AM SSB)

Question 1: Calculate the following values for the single-tone message signal:

**peak power
average power
baseband bandwidth**

Answer: peak power = $P_p = \frac{A_p^2}{2} \Rightarrow 15^2 / 2 \Rightarrow 112.5$

average power = $P = A_0^2 + \frac{1}{2} \sum_{N=1}^{\infty} (A_N^2 + B_N^2) \Rightarrow 15^2 / 2 \Rightarrow 112.5$

baseband bandwidth = 130 Hz

Question 2: Predict the following values for the single-tone AM SSB signal:

**peak power
average power
bandwidth**

Answer: peak power = $P_p = \frac{A_p^2}{2} \Rightarrow 7.5^2 / 2 \Rightarrow 28.125$

average power = $P = \frac{A^2}{2} \Rightarrow 7.5^2 / 2 \Rightarrow 28.125$

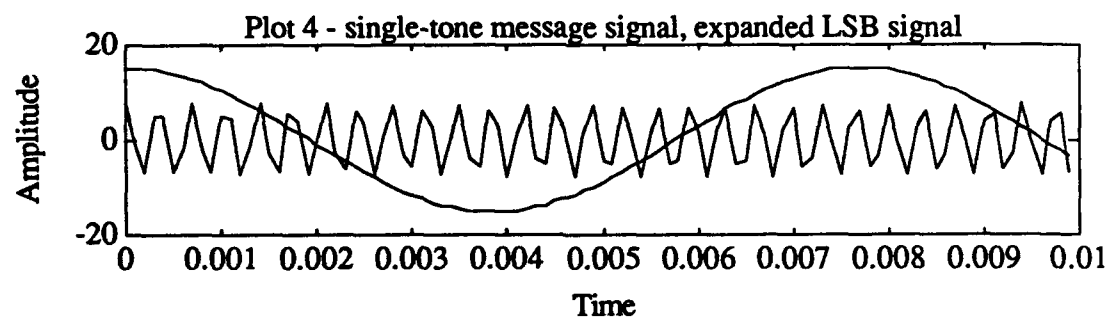
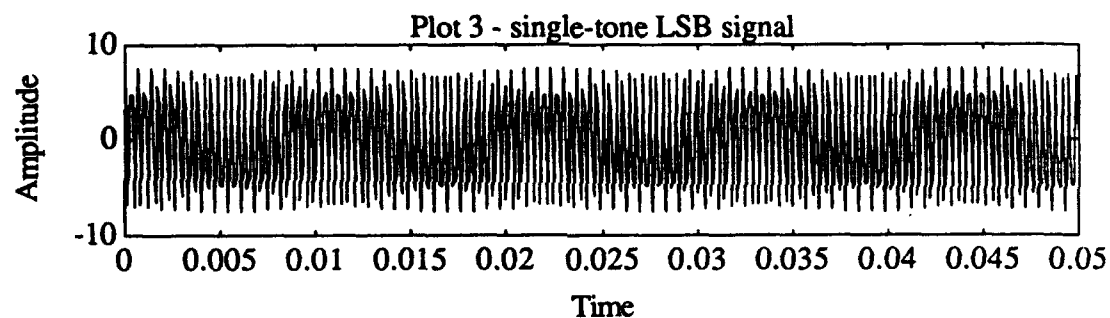
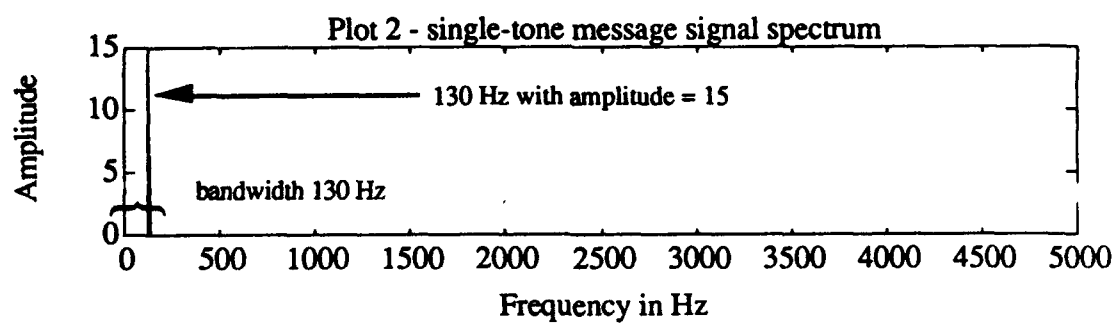
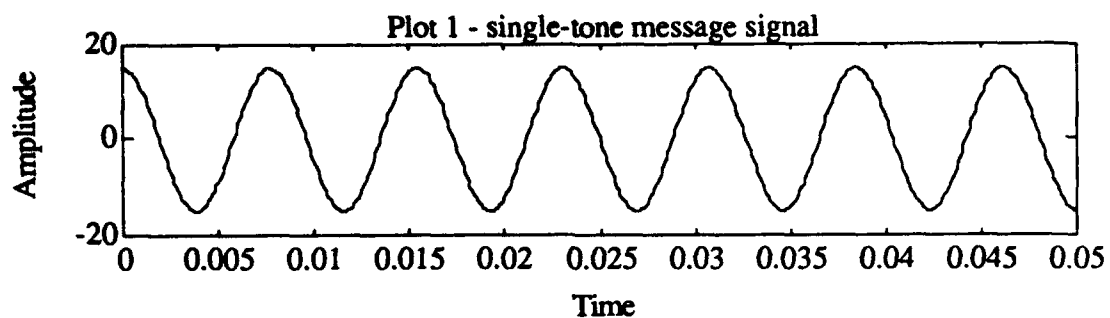
baseband bandwidth = 130 Hz

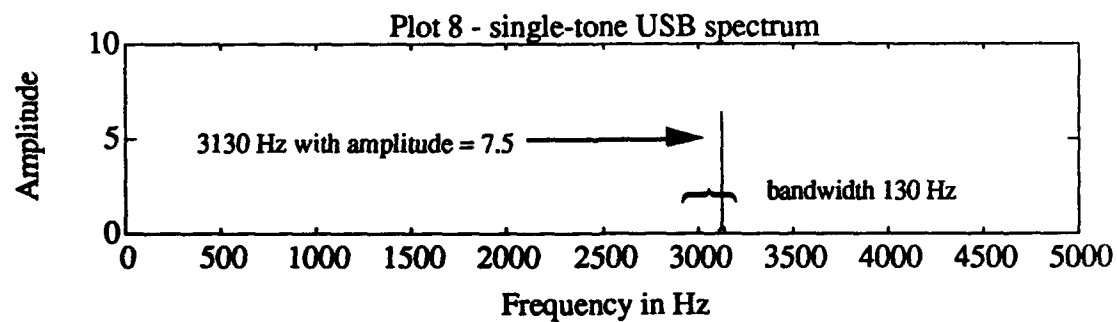
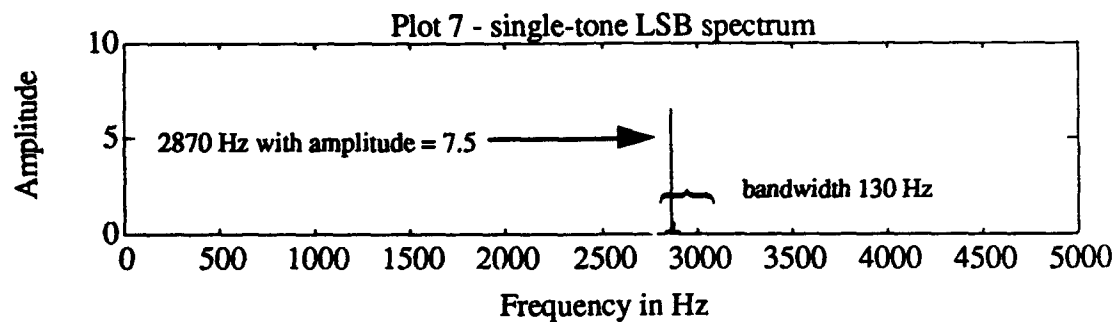
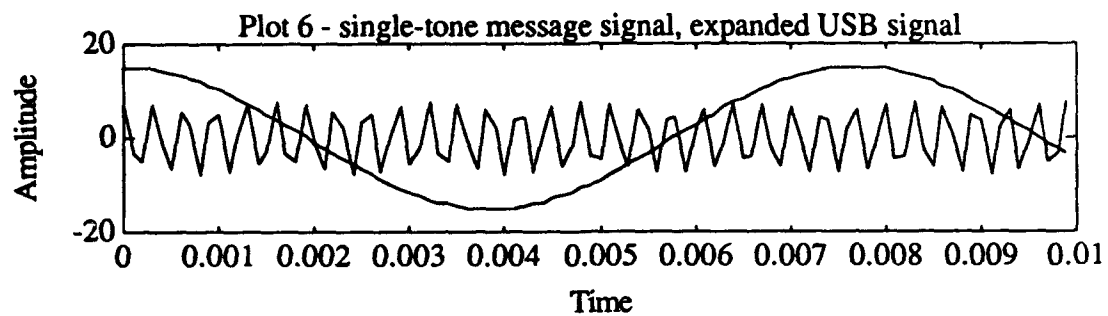
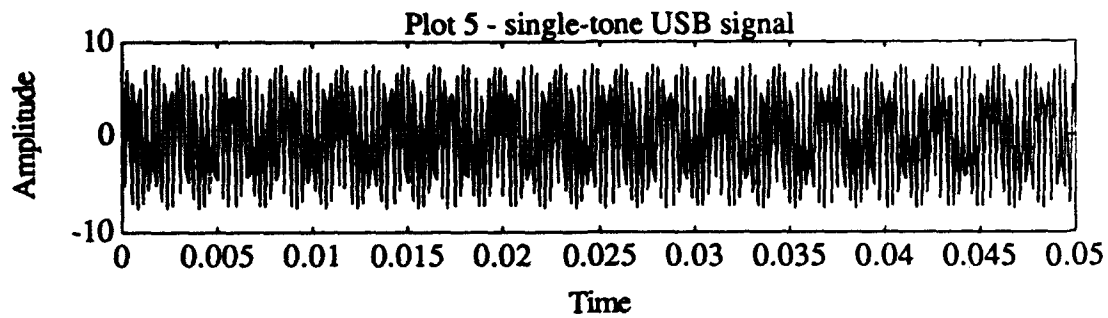
Question 3: From Plots 9 and 10, obtain the values representing peak and average power for the signal-tone signal, and record them.

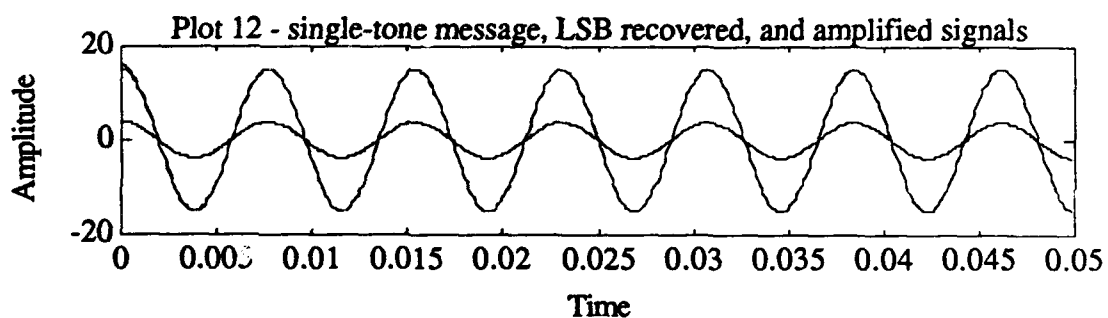
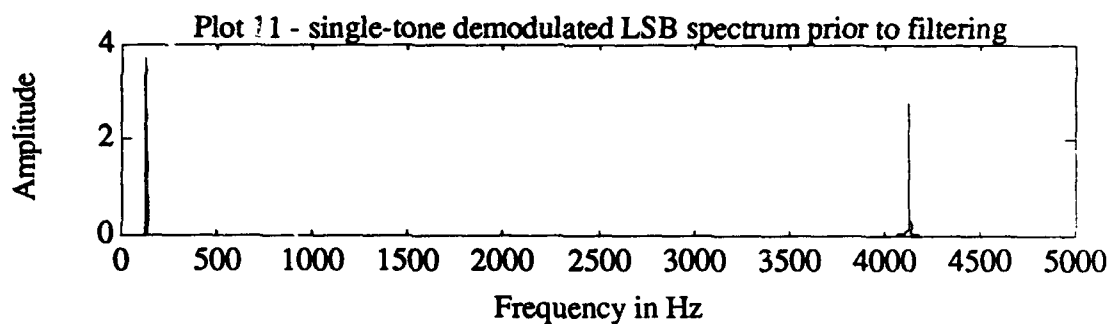
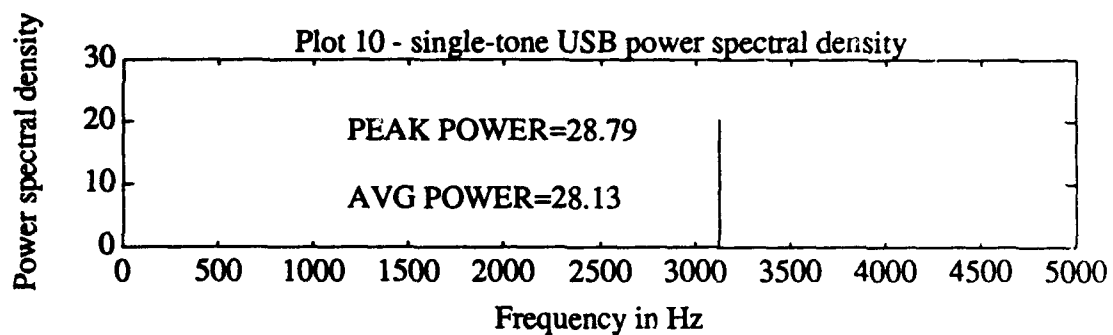
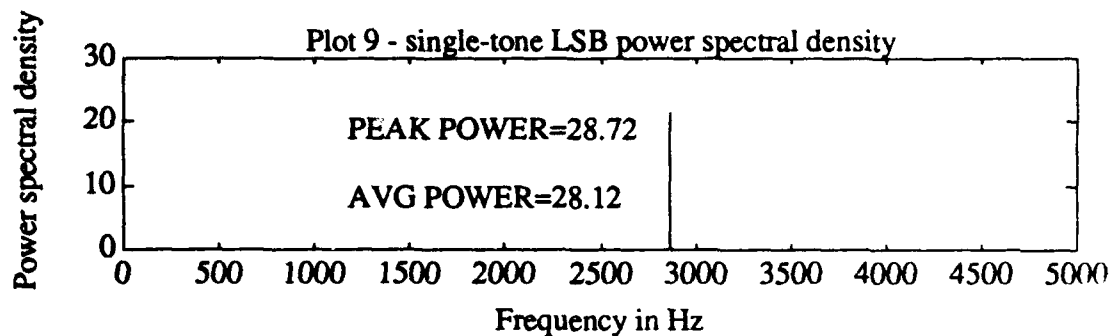
Do your calculations for bandwidth and power agree with the computer-generated values and spectrum?

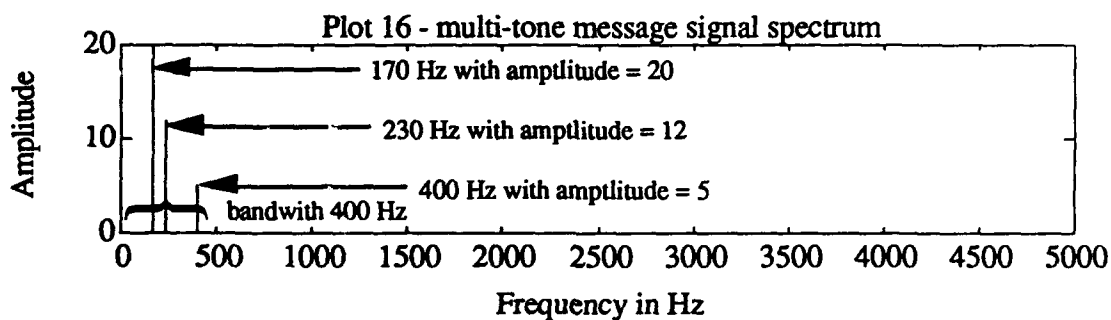
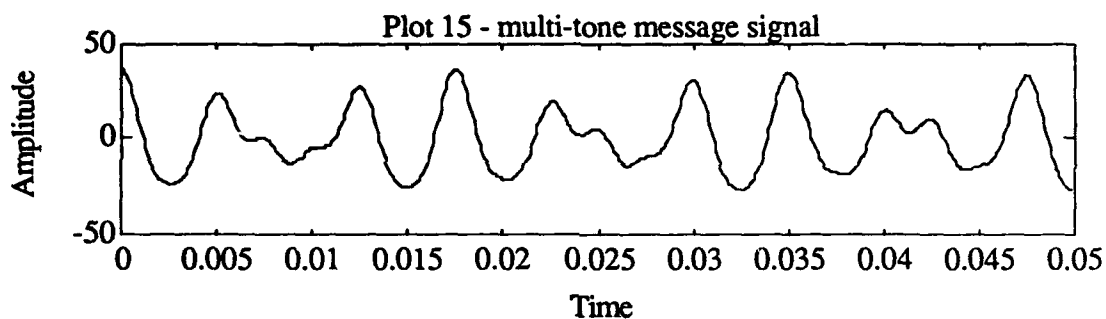
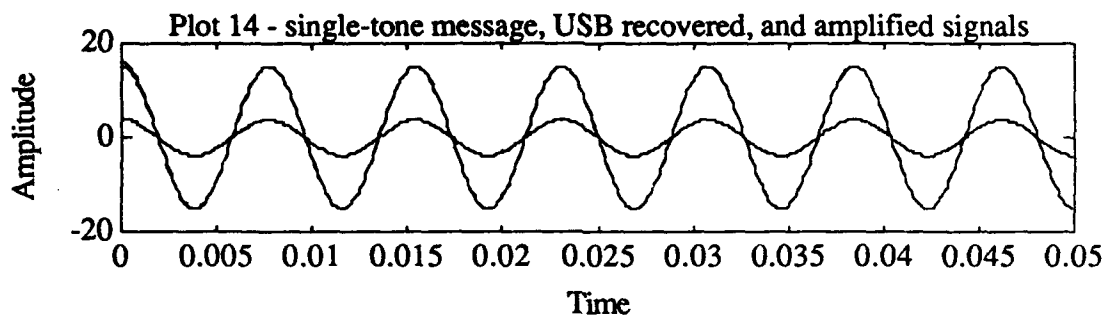
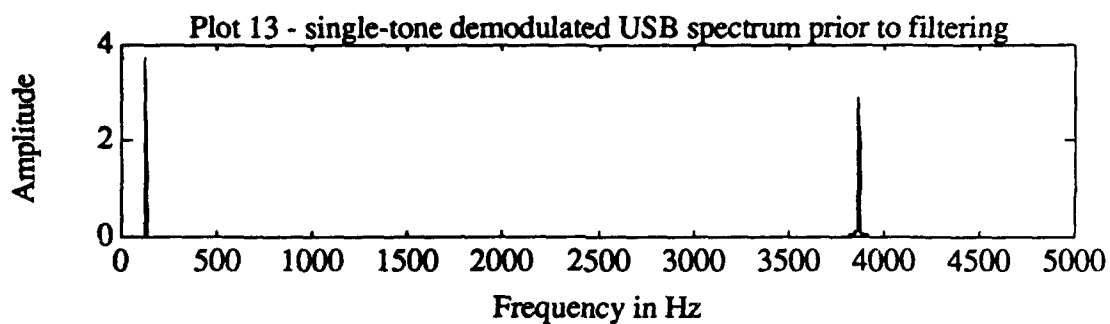
Answer: single-tone LSB peak power = 28.72
single-tone USB peak power = 28.12
single-tone LSB average power = 28.79
single-tone USB average power = 28.13

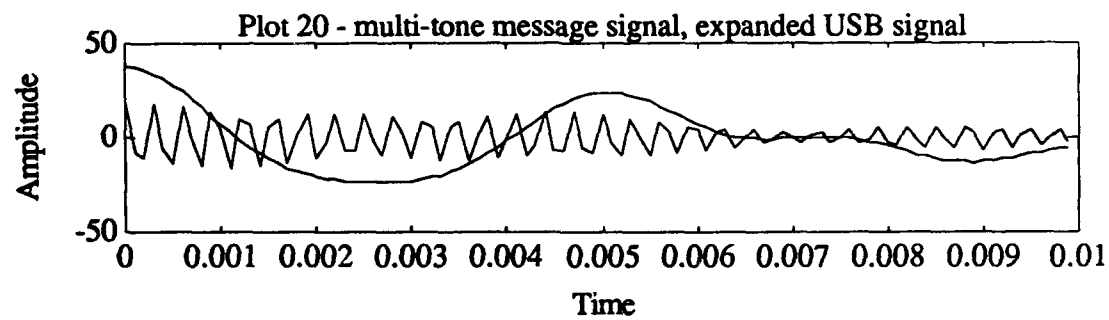
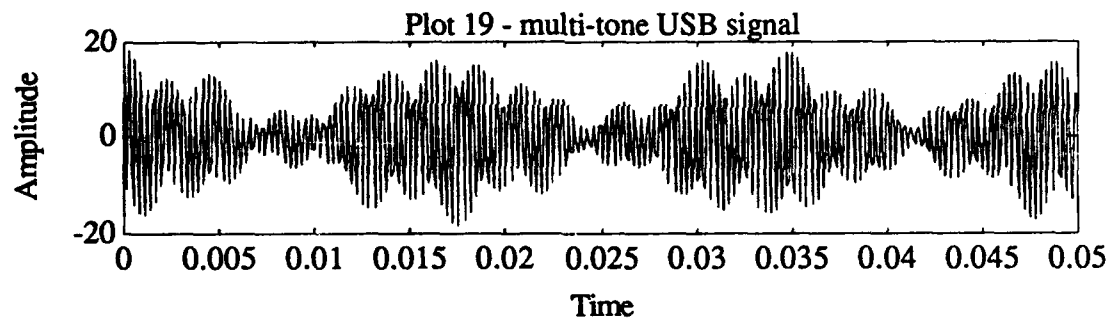
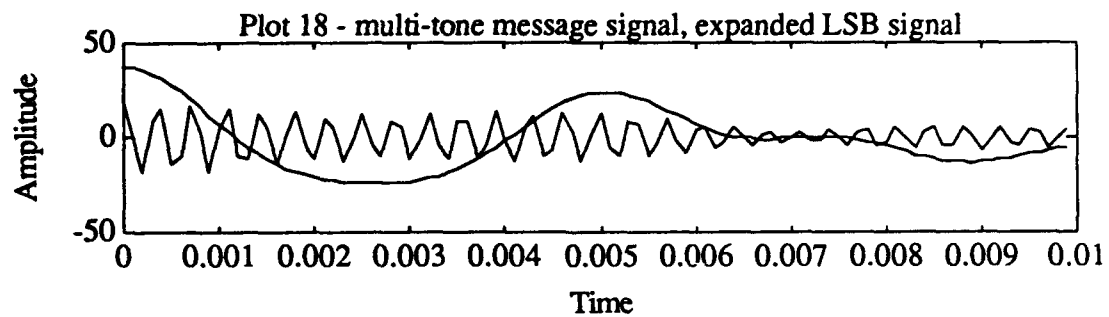
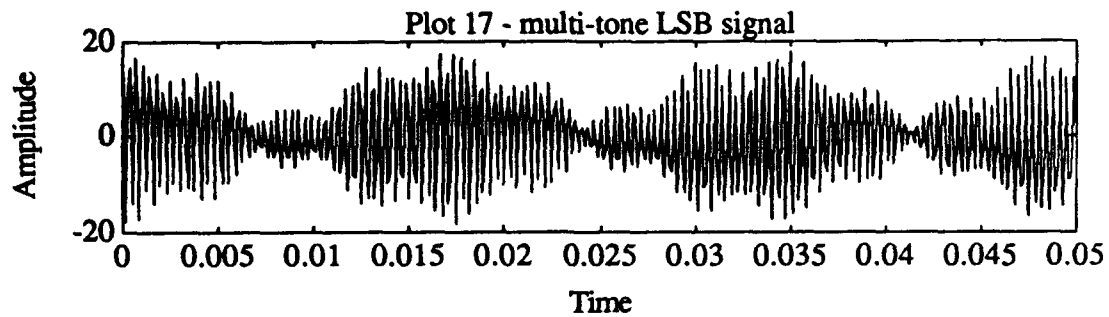
Yes—calculations agree.

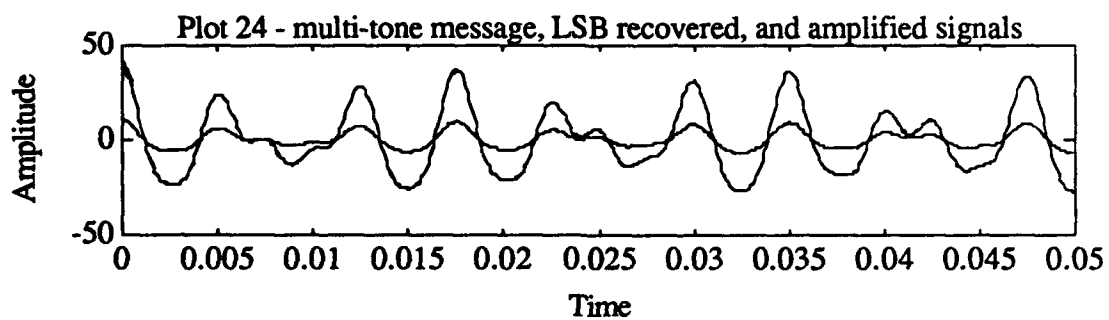
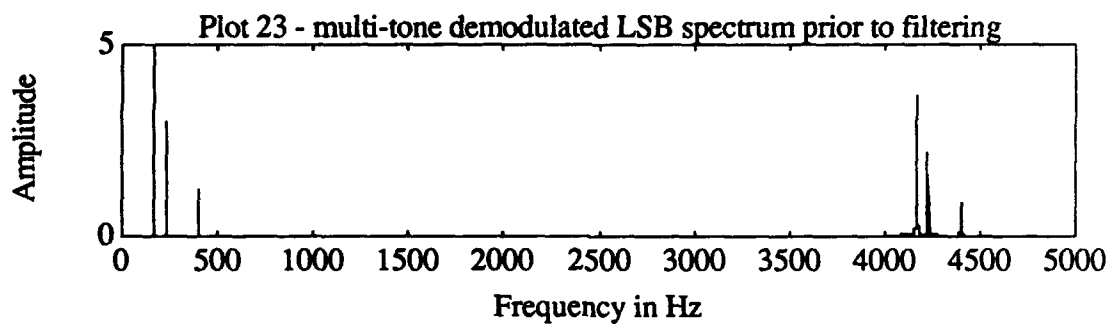
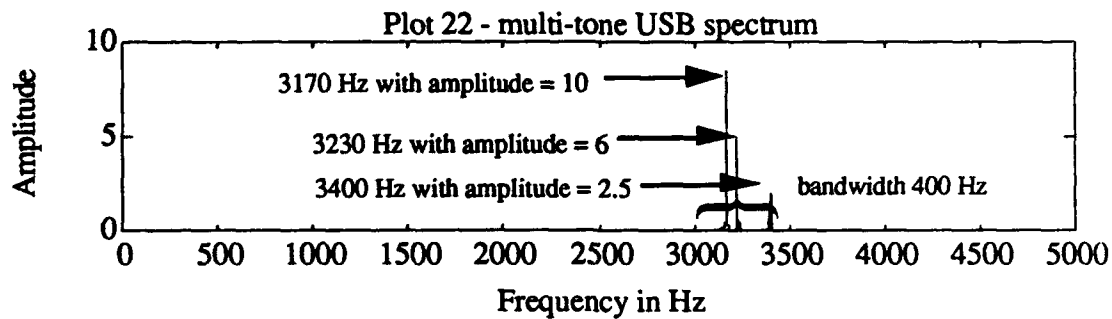
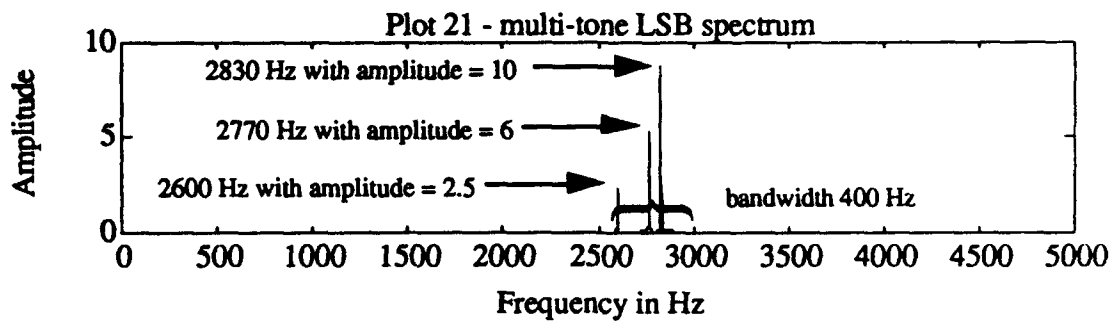


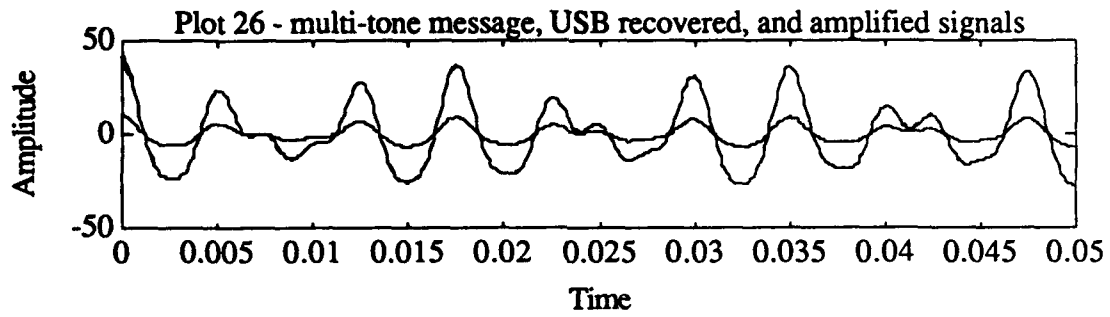
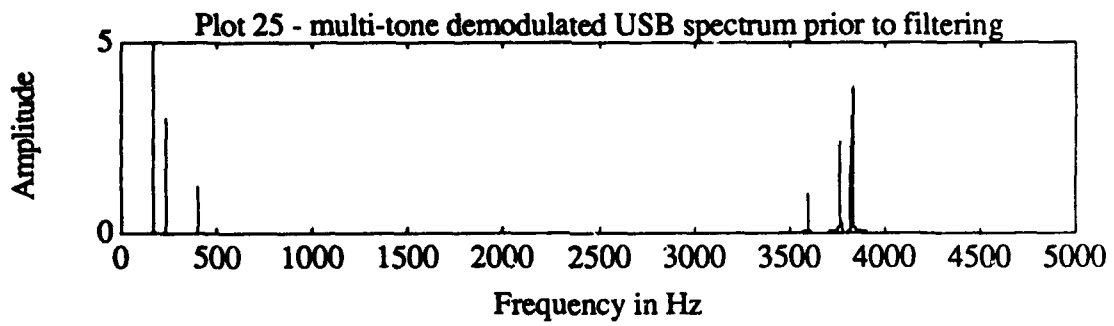












lab6scr.m

%Computer-aided Lab 6 script for student use

%%
%Computer-aided Lab 6 Amplitude Modulation AM SSB
%%
%Part 1--Observe the AM SSB modulation process with single-tone
% input
%A. Generating the signal and spectrum

clear
clg

delta_t=.0001;
t=0:delta_t:1; %time vector
s=15*cos(2*pi*130*t); %single-tone signal

fc=3000; %modulating frequency for the carrier signal
cutoff=150; %ideal lowpass filter cutoff frequency for single-tone

%Plot 1

subplot(211), %plot the signal
plot(t(1:500),s(1:500))
title('Plot 1 - single-tone message signal')
xlabel('Time')
ylabel('Amplitude')

[specs,HZ]=spectral(s,delta_t); %generate the spectrum

%Plot 2

subplot(212), %plot the spectrum
plot(HZ,specs)
title('Plot 2 - single-tone message signal spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

clear specs;

%B. Observe the single-tone AM SSB modulated
% signals and spectra

[lsbs,usbs]=ssb(t,s,1,fc); %generate the upper and lower sideband signals

%Plot 3

```
subplot(211), %plot the lower sideband signal
plot(t(1:500),lsbs(1:500));
title('Plot 3 - single-tone LSB signal')
xlabel('Time')
ylabel('Amplitude')
```

%Plot 4

```
subplot(212), %plot detailed view--no phase shifts
plot(t(1:100),[s(1:100);lsbs(1:100)])
title('Plot 4 - single-tone message signal, expanded LSB signal')
xlabel('Time')
ylabel('Amplitude')
```

```
pause
clg
```

%Plot 5

```
subplot(211), %plot the upper sideband signal
plot(t(1:500),usbs(1:500));
title('Plot 5 - single-tone USB signal')
xlabel('Time')
ylabel('Amplitude')
```

%Plot 6

```
subplot(212), %plot detailed view--no phase shifts
plot(t(1:100),[s(1:100);usbs(1:100)])
title('Plot 6 - single-tone message signal, expanded USB signal')
xlabel('Time')
ylabel('Amplitude')
```

```
pause
clg
```

```
[specslbs,HZ]=spectral(lsbs,delta_t); %generate the lower sideband spectrum
```

%Plot 7

```
subplot(211), %plot the lower sideband spectrum
plot(HZ,specslbs);
title('Plot 7 - single-tone LSB spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')
```



```
specusbs=spectral(usbs,delta_t); %generate the upper sideband spectrum
```

```
%Plot 8
```

```
subplot(212), %plot the upper sideband spectrum  
plot(Hz,specusbs);  
title('Plot 8 - single-tone USB spectrum')  
xlabel('Frequency in Hz')  
ylabel('Amplitude')
```

```
clear specusbs;clear specsbs;
```

```
pause  
clg
```

```
%C. Verify the power and bandwidth of the SSB signals
```

```
lsb_pk_pwr_sngl=((max(lsb))^2)/2 %find the peak power
```

```
[psdlb,HZ]=psd(lsb,delta_t); %generate the lower sideband power  
%spectral density
```

```
lsb_avg_pwr_sngl=sum(psdlb) %find average power by summing the power  
%spectral densities
```

```
%Plot 9
```

```
subplot(211), %plot the lower sideband power spectral density  
plot(HZ,psdlb)  
title('Plot 9 - single-tone LSB power spectral density')  
xlabel('Frequency in Hz')  
ylabel('Power spectral density')
```

```
clear psdlb;
```

```
usb_pk_pwr_sngl=((max(usbs))^2)/2 %find the peak power
```

```
psdusb=psd(usbs,delta_t); %generate the upper sideband power  
%spectral density
```

```
usb_avg_pwr_sngl=sum(psdusb) %find average power by summing the power  
%spectral densities
```

```
%Plot 10
```

```
subplot(212), %plot the upper sideband power spectral density  
plot(HZ,psdusb)  
title('Plot 10 - single-tone USB power spectral density')  
xlabel('Frequency in Hz')
```

```

ylabel('Power spectral density')

clear psdusb;

pause
clg

%D. Observe the recovery of the SSB signals

demodlsbs=cos(2*pi*fc*t).*lsbs; %recover the signal by multiplying by
                                %the carrier
clear lsbs;
[recspeclsbs,HZ,fflsbs]=spectral(demodlsbs,delta_t);
clear demodlsbs;

%Plot 11

subplot(211), %plot the remodulated lsb spectrum
plot(HZ,recspeclsbs)
title('Plot 11 - single-tone demodulated LSB spectrum prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear recspeclsbs;

reclsbs=recoverm(fflsbs,'ideallow',HZ,cutoff); %recover and filter
clear fflsbs;
bigreclsbs=reclsbs*4; %amplify signal

%Plot 12

subplot(212), %plot the recovered signal on top of the message signal
plot(t(1:500),[s(1:500);reclsbs(1:500)])
title('Plot 12 - single-tone message, LSB recovered, and amplified signals')
xlabel('Time')
ylabel('Amplitude')

hold on
pause
plot(t(1:500),bigreclsbs(1:500),'b')
hold off

pause
clg

demodusbs=cos(2*pi*fc*t).*usbs; %recover the signal by multiplying by
                                %the carrier
clear usbs;

```

```

[recspecusbs,Hz,fftusbs]=spectral(demodusbs,deltat);
clear demodusbs;

%Plot 13

subplot(211), %plot the remodulated usb spectrum
plot(Hz,recspecusbs)
title('Plot 13 - single-tone demodulated USB spectrum prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear recspecusbs;

recusbs=recoverm(fftusbs,'ideallow',Hz,cutoff); %recover and filter
clear fftusbs;
bigrecusbs=recusbs*4; %amplify signal

%Plot 14

subplot(212), %plot the recovered signal on top of the message signal
plot(t(1:500),[s(1:500);recusbs(1:500)])
title('Plot 14 - single-tone message, USB recovered, and amplified signals')
xlabel('Time')
ylabel('Amplitude')

hold on
pause
plot(t(1:500),bigrecusbs(1:500),'b')
hold off

pause

%%%%%%%%%%%%%%
%Part 2--Observe the AM SSB modulation process with multi-tone
%    input
%A. Generating the signal and spectrum

clear
clg

deltat=.0001;
t=0:deltat:1; %time vector
%multi-tone signal
s=5*cos(2*pi*400*t)+12*cos(2*pi*230*t)+20*cos(2*pi*170*t);

fc=3000; %modulating frequency for the carrier signal
cutoff=420; %ideal lowpass filter cutoff frequency for multi-tone

```

%Plot 15

```
subplot(211), %plot the signal
plot(t(1:500),s(1:500))
title('Plot 15 - multi-tone message signal')
xlabel('Time')
ylabel('Amplitude')
```

```
[specs,Hz]=spectral(s,delta_t); %generate the spectrum
```

%Plot 16

```
subplot(212), %plot the spectrum
plot(Hz,specs)
title('Plot 16 - multi-tone message signal spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')
```

```
pause
clg
```

```
clear specs;
```

```
%B. Observe the multi-tone AM SSB modulated
% signals and spectra
```

```
[lsbs,usbs]=ssb(t,s,1,fc); %generate the upper and lower sideband signals
```

%Plot 17

```
subplot(211), %plot the lower sideband signal
plot(t(1:500),lsbs(1:500));
title('Plot 17 - multi-tone LSB signal')
xlabel('Time')
ylabel('Amplitude')
```

%Plot 18

```
subplot(212), %plot detailed view--no phase shifts
plot(t(1:100),[s(1:100);lsbs(1:100)])
title('Plot 18 - multi-tone message signal, expanded LSB signal')
xlabel('Time')
ylabel('Amplitude')
```

```
pause
clg
```

%Plot 19

```

subplot(211), %plot the upper sideband signal
plot(t(1:500),usbs(1:500));
title('Plot 19 - multi-tone USB signal')
xlabel('Time')
ylabel('Amplitude')

%Plot 20

subplot(212), %plot detailed view--no phase shifts
plot(t(1:100),[s(1:100);usbs(1:100)])
title('Plot 20 - multi-tone message signal, expanded USB signal')
xlabel('Time')
ylabel('Amplitude')

pause
clg

[specslsbs,HZ]=spectral(lsb, delta_t); %generate the lower sideband spectrum

%Plot 21

subplot(211), %plot the lower sideband spectrum
plot(HZ,specslsbs);
title('Plot 21 - multi-tone LSB spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

specusbs=spectral(usbs,delta_t); %generate the upper sideband spectrum

%Plot 22

subplot(212), %plot the upper sideband spectrum
plot(HZ,specusbs);
title('Plot 22 - multi-tone USB spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear specusbs;clear specslsbs;

pause
clg

%C. Observe the recovery of the SSB signals

demodlsbs=cos(2*pi*fc*t).*lsbs; %recover the signal by multiplying by
                                %the carrier
clear lsbs;

```

```

[recspeclsbs,HZ,fftlsbs]=spectral(demodlsbs,delta_t);
clear demodlsbs;

%Plot 23

subplot(211), %plot the remodulated lsb spectrum
plot(HZ,recspeclsbs)
title('Plot 23 - multi-tone demodulated LSB spectrum prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear recspeclsbs;

recclsbs=recoverm(fftlsbs,'ideallow',HZ,cutoff); %recover and filter
clear fftlsbs;
bigrecclsbs=recclsbs*4; %amplify signal

%Plot 24

subplot(212), %plot the recovered signal on top of the message signal
plot(t(1:500),[s(1:500);recclsbs(1:500)])
title('Plot 24 - multi-tone message, LSB recovered, and amplified signals')
xlabel('Time')
ylabel('Amplitude')

hold on
pause
plot(t(1:500),bigrecclsbs(1:500),'b')
hold off

pause
clg

demodusbs=cos(2*pi*fc*t).*usbs; %recover the signal by multiplying by
                                %the carrier
clear usbs;
[recspecusbs,HZ,fftusbs]=spectral(demodusbs,delta_t);
clear demodusbs;

%Plot 25

subplot(211), %plot the remodulated usb spectrum
plot(HZ,recspecusbs)
title('Plot 25 - multi-tone demodulated USB spectrum prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear recspecusbs;

```

```

recusbs=recoverm(ffusbs,'ideal',Hz,cutoff); %recover and filter
clear ffusbs;
bigrecusbs=recusbs*4; %amplify signal

%Plot 26

subplot(212), %plot the recovered signal on top of the message signal
plot(t(1:500),[s(1:500);recusbs(1:500)])
title('Plot 26 - multi-tone message, USB recovered, and amplified signals')
xlabel('Time')
ylabel('Amplitude')

hold on
pause
plot(t(1:500),bigrecusbs(1:500),'b')
hold off

```

EO 3513 Computer-aided Laboratory 7 Key

Conventional Amplitude Modulation

(Conventional AM)

Question 1: Calculate the following values for the single-tone message signal:

peak power
average power
baseband bandwidth

Answer: peak power = $P_p = \frac{A_p^2}{2} \Rightarrow 1^2 / 2 \Rightarrow 0.5$

average power = $P = A_0^2 + \frac{1}{2} \sum_{N=1}^{\infty} (A_N^2 + B_N^2) \Rightarrow 1^2 / 2 \Rightarrow 0.5$

bandwidth = 150 Hz

Question 2: Predict the following values for the single-tone conventional AM signal:

peak power
average power
bandwidth

Answer: peak power = $P_p = (1 + m)^2 P_c \Rightarrow (1 + 0.8)^2 * 0.5 \Rightarrow 1.62$

average power = $P = \left(1 + \frac{m^2}{2}\right) P_c \Rightarrow (1 + (0.8^2 / 2)) * 0.5 \Rightarrow 0.66$

bandwidth = 300 Hz

Question 3: From Plot 5, obtain the values representing peak and average power for the single-tone conventional AM signal, and record them.

Do your calculations for bandwidth and power agree with the computer-generated values and spectrum?

Answer: peak power = 1.6172
average power = 0.6598

Yes—calculations agree.

Question 4: Refer to Plot 5 and estimate the percentage of power contained in the carrier. What might be an advantage of having this amount of power transmitted in the carrier as opposed to transmission in the sidebands?

Answer: When $m = 1$, approximately 33% of the total average power is carried in the sidebands, making the percentage of power in the carrier approximately 67%. The signal shown in Plot 5 had a value of $m = 0.8$. Approximately 75% of the power appears to be transmitted in the carrier.

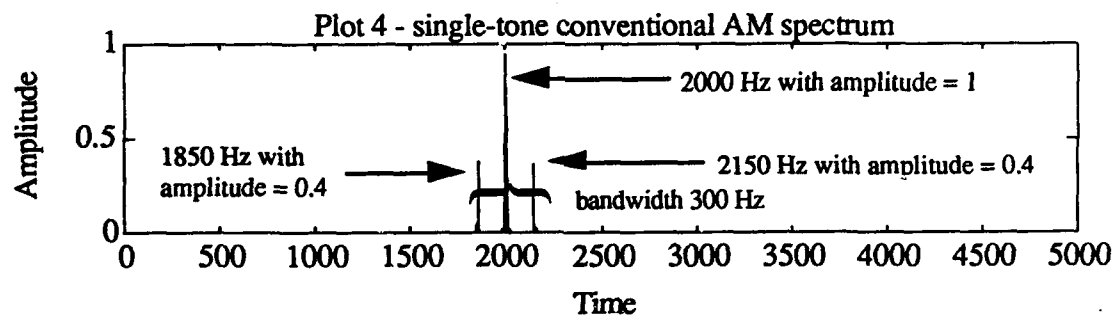
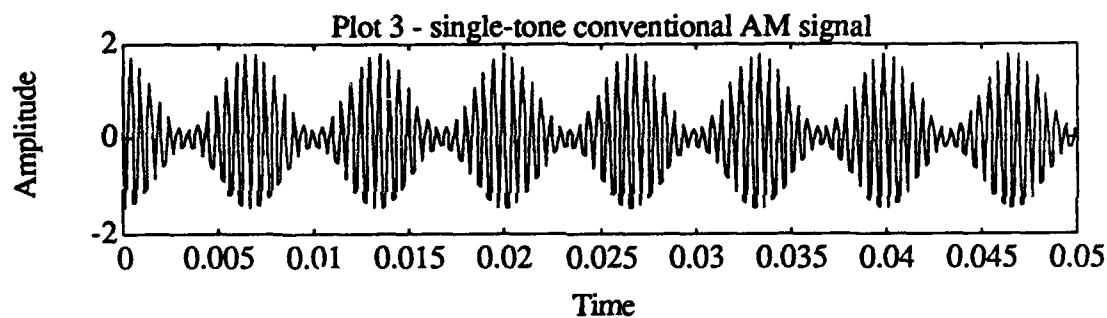
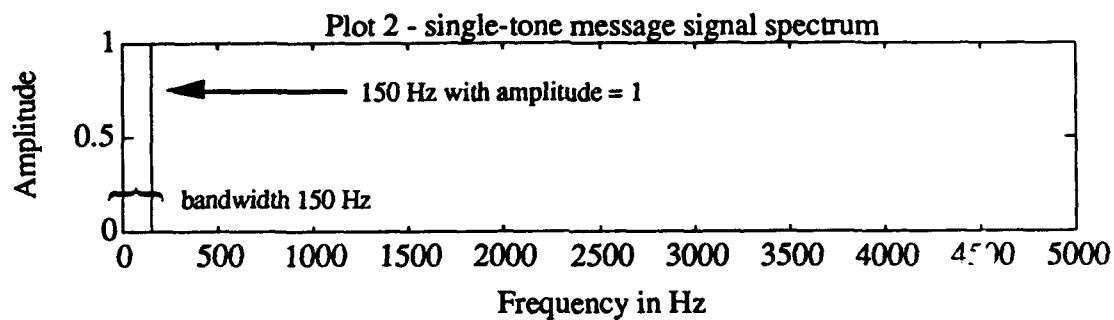
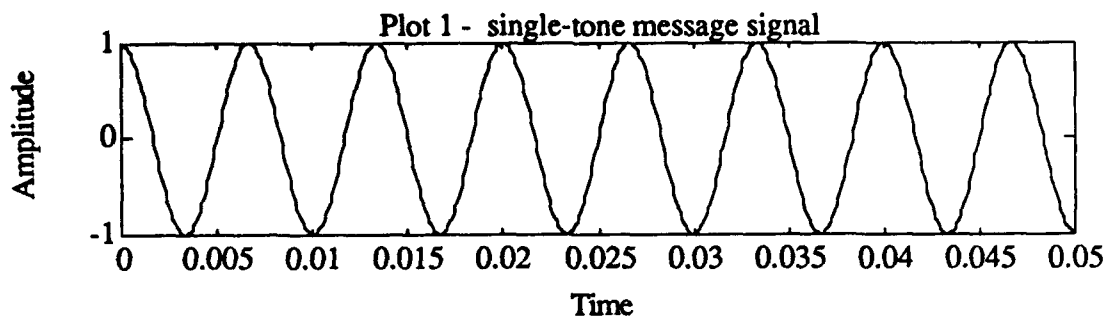
Transmitting a high percentage of power in the carrier makes the carrier easier to detect.

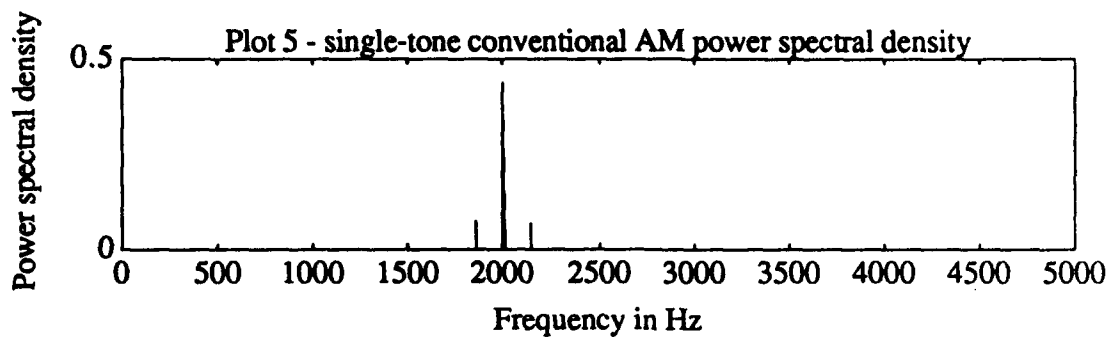
Question 5: What result of overmodulation prevents the use of an envelope detector for the conventional AM signal?

Answer: The presence of phase shifts in the overmodulated conventional AM signal prevents use of an envelope detector.

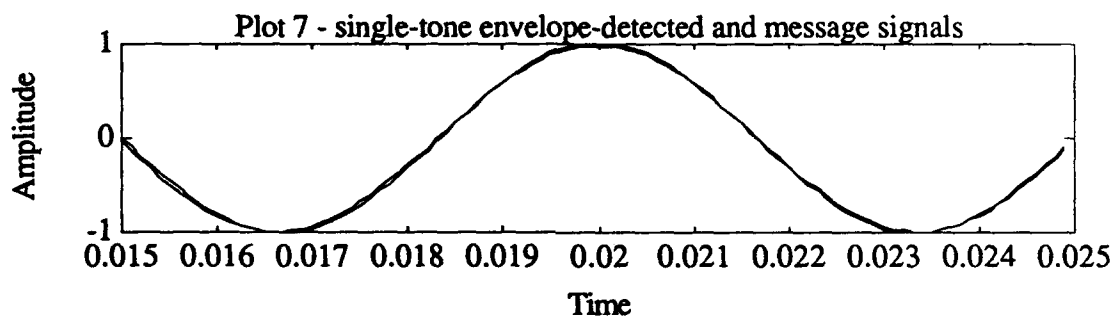
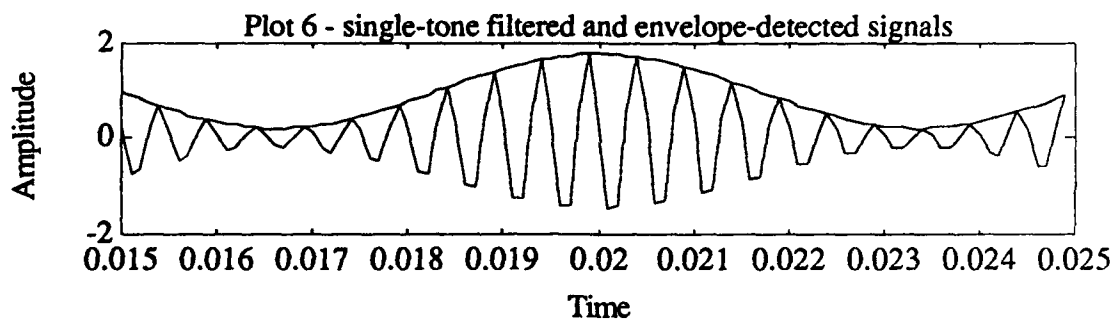
Question 6: What type of detection is needed for an overmodulated conventional AM signal? Why?

Coherent detection (detection using the carrier) is necessary for an overmodulated conventional AM signal. The phase shifts preclude envelope detection.

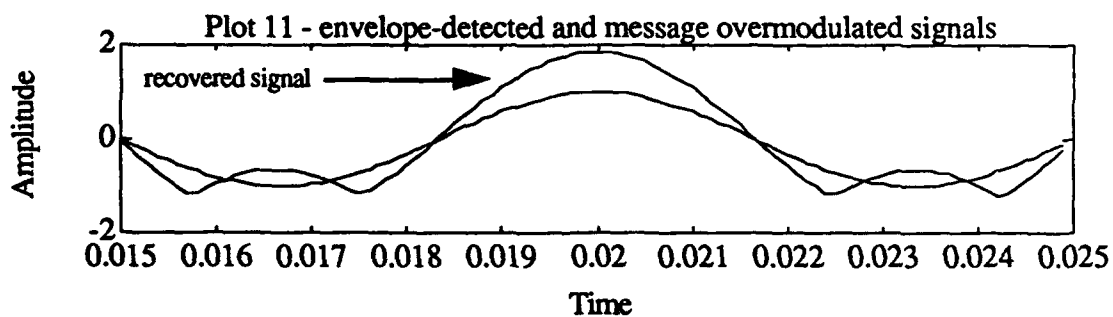
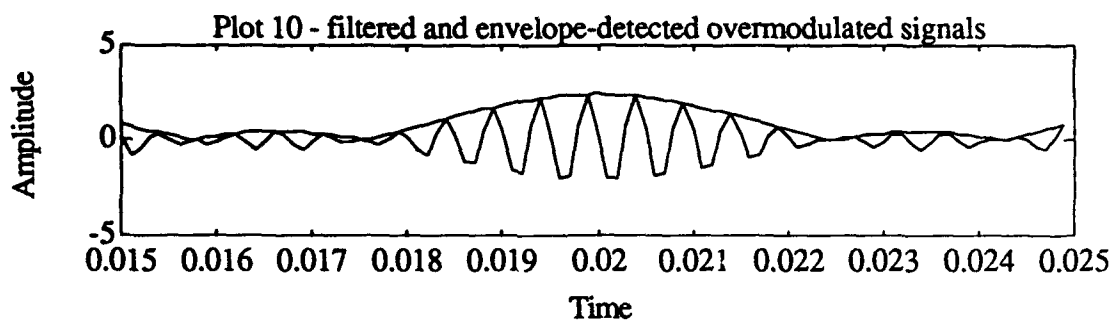
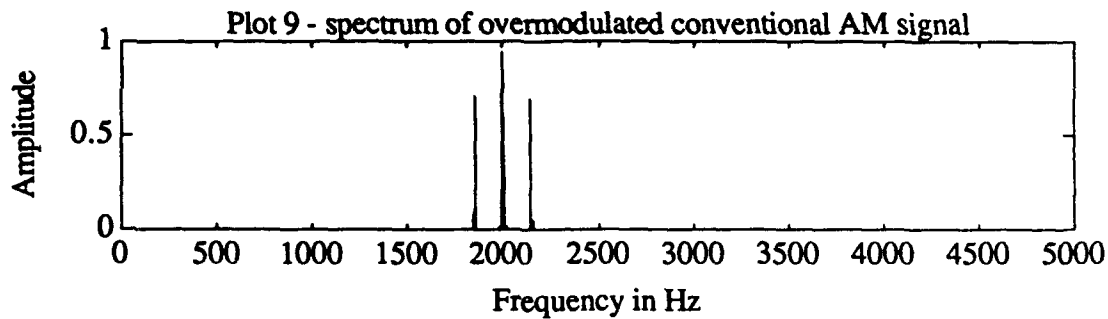
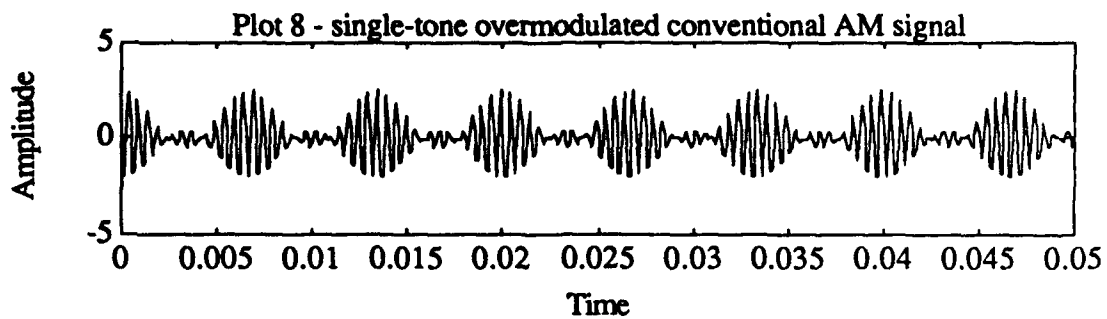


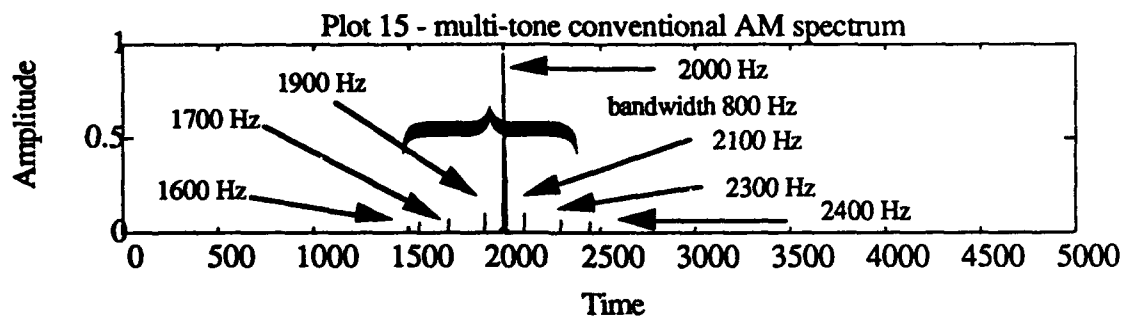
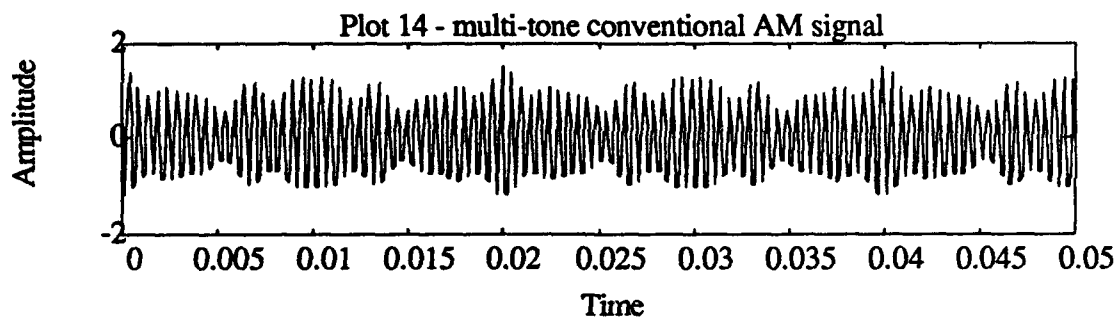
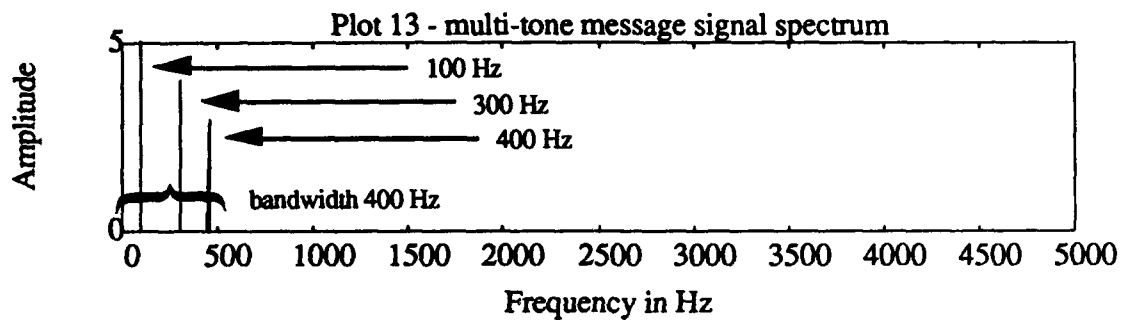
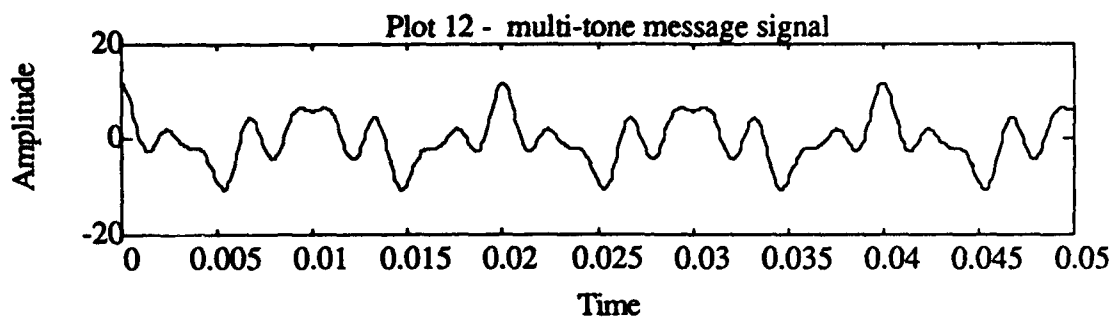


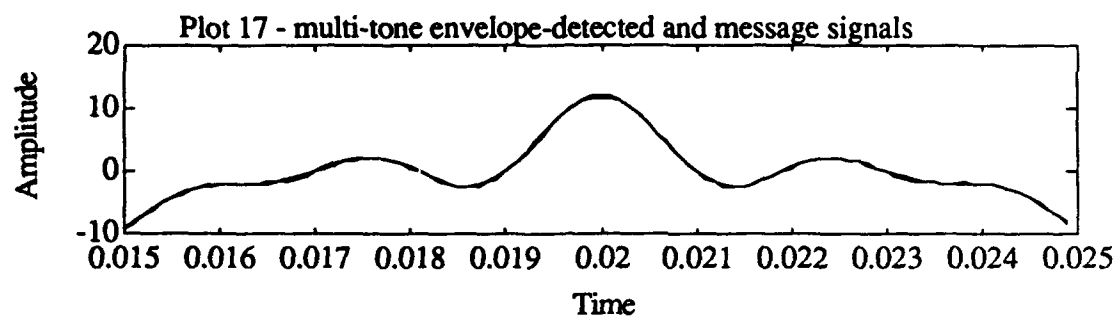
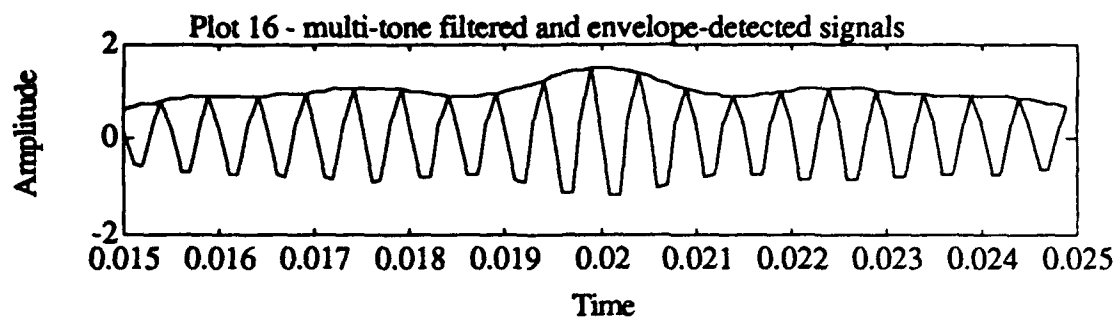
NO PLOT HERE--JUST PRESS RETURN



Computer-aided Laboratory 7 Key—page 4







lab7scr.m

%Computer-aided Lab 7 script for student use

%%%

%Computer-aided Lab 7 Conventional Amplitude Modulation--

% Conventional AM

%%%

%PART 1--Observe the conventional AM modulation process

% using single-tone input

%A. Generating the signal and spectrum

clear

clg

delta_t=.0001;

t=0:delta_t:1; %time vector

s=cos(2*pi*150*t); %single-tone signal

fc=2000; %modulating frequency for the carrier signal

fs=150; %highest frequency in the message signal

m=.8; %conventional AM modulation index

over_m=1.5; %index for overmodulated signal

cutoff1=1800; %lower cutoff frequency for ideal bandpass filter

%for single-tone signal

cutoff2=2200; %upper cutoff frequency for ideal bandpass filter

%for single-tone signal

%Plot 1

subplot(211), %plot the signal

plot(t(1:500),s(1:500))

title('Plot 1 - single-tone message signal')

xlabel('Time')

ylabel('Amplitude')

[specs,HZ]=spectral(s,delta_t); %generate the spectrum

%Plot 2

subplot(212), %plot the spectrum

plot(HZ,specs,'g')

title('Plot 2 - single-tone message signal spectrum')

xlabel('Frequency in Hz')

ylabel('Amplitude')

pause

Computer-aided Laboratory 7 Key—page 8

clg

%B. Observe the conventional AM signal and spectrum

convams=conv_am(s,delta_t,fc,m);

%Plot 3

subplot(211), %plot the conventional AM modulated signal

plot(t(1:500),convams(1:500))

title('Plot 3 - single-tone conventional AM signal')

xlabel('Time')

ylabel('Amplitude')

[convamspec,HZ,fftconvams]=spectral(convams,delta_t); %generate the modulated
%spectrum

%Plot 4

subplot(212), %plot the modulated spectrum

plot(HZ,convamspec,'b')

title('Plot 4 - single-tone conventional AM spectrum')

xlabel('Time')

ylabel('Amplitude')

pause

clg

%C. Verify the power and bandwidth of the conventional AM signal

cam_peak_power=((max(convams))^2)/2; %find the peak power

psdcam=psd(convams,delta_t); %generate the power spectral density

cam_avg_power=sum(psdcam); %find average power by summing the power
%spectral density values

str1=num2str(cam_peak_power);

str2=num2str(cam_avg_power);

%Plot 5

subplot(211), %plot the power spectral density for the modulated signal

plot(HZ,psdcam)

title('Plot 5 - single-tone conventional AM power spectral density')

xlabel('Frequency in Hz')

ylabel('Power spectral density')

text(.5,.8,['PEAK POWER=' str1],'sc')

text(.5,.7,['AVG POWER=' str2],'sc')


```

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

pause
clg

%D. Recover and detect the conventional AM signal

    %recover and filter
filsig=recoverm(fficonvams,'idealbnd',Hz,cutoff1,cutoff2);
filsig=filsig(1:300); %shorten the vector for speed
envsig=envelope(filsig); %envelope detect the signal

bigsig=(envsig-1)/m; %remove DC value and divide by m

%Plot 6    %plot the envelope detected signal over the
           %recovered signal
subplot(211),
plot(t(151:250),[filsig(151:250);envsig(151:250)])
title('Plot 6 - single-tone filtered and envelope-detected signals')
xlabel('Time')
ylabel('Amplitude')

           %plot the message signal over the
           %amplified envelope-detected signal
%Plot 7

subplot(212),
plot(t(151:250),bigsig(151:250))
title('Plot 7 - single-tone envelope-detected and message signals')
xlabel('Time')
ylabel('Amplitude')
hold on
pause
plot(t(151:250),s(151:250), 'b')
hold off

pause
clg

%%%%%%%%%%%%%%
%PART 2--Observe the effect of overmodulating the
%    conventional AM signal
%A. Observe the overmodulated conventional AM signal

clear
clg

```

```

delta_t=.0001;
t=0:delta_t:1; %time vector
s=cos(2*pi*150*t); %single-tone signal

fc=2000; %modulating frequency for the carrier signal
fs=150; %highest frequency in the message signal
m=.8; %conventional AM modulation index
over_m=1.5; %index for overmodulated signal

cutoff1=1800; %lower cutoff frequency for ideal bandpass filter
           %for single-tone signal
cutoff2=2200; %upper cutoff frequency for ideal bandpass filter
           %for single-tone signal

oconvams=conv_am(s,delta_t,fc,over_m); %overmodulate the signal

%Plot 8

subplot(211), %plot the overmodulated signal
plot(t(1:500),oconvams(1:500))
title('Plot 8 - single-tone overmodulated conventional AM signal')
xlabel('Time')
ylabel('Amplitude')

[oconvamspec,HZ,fftconvams]=spectral(oconvams,delta_t); %generate the modulated
                                                    %spectrum

%Plot 9

subplot(212), %plot the spectrum of the overmodulated signal
plot(HZ,oconvamspec,'g')
title('Plot 9 - spectrum of overmodulated conventional AM signal')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

%B. Observe the effect of overmodulation on recovery

filtsig=recoverm(fftconvams,'idealbnd',HZ,cutoff1,cutoff2);
filtsig=filtsig(1:300); %shorten the vector for speed
envsig=envelope(filtsig); %use envelope detector
bigsig=((envsig-1)/m); %remove DC value, divide by m

%Plot 10 %plot the envelope detected signal over the
          %recovered signal
subplot(211),

```

```

plot(t(151:250),[filtsig(151:250);envsig(151:250)])
title('Plot 10 - filtered and envelope-detected overmodulated signals')
xlabel('Time')
ylabel('Amplitude')

```

```

    %plot the message signal over the
    %amplified envelope-detected signal

```

```

%Plot 11

```

```

subplot(212),
plot(t(151:250),bigsig(151:250))
title('Plot 11 - envelope-detected and message overmodulated signals')
xlabel('Time')
ylabel('Amplitude')
hold on
pause
plot(t(151:250),s(151:250), 'b')
hold off

```

```

pause
clg

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PART 3--Observe the conventional AM modulation process
%    using multi-tone input
%A. Generating the signal and spectrum

```

```

clear

```

```

delta_t=.0001;
t=0:delta_t:1; %time vector
s=5*cos(2*pi*100*t)+4*cos(2*pi*300*t)+3*cos(2*pi*450*t); %single-tone signal
max_s=max(s); %save value to expand signal during detection

```

```

fc=2000; %modulating frequency for the carrier signal
fs=450; %highest frequency in the message signal
m=.5; %conventional AM modulation index

```

```

cutoff1=1500; %lower cutoff frequency for ideal bandpass filter
    %for multi-tone signal
cutoff2=2500; %upper cutoff frequency for ideal bandpass filter
    %for multi-tone signal

```

```

%Plot 12

```

```

subplot(211), %plot the signal
plot(t(1:500),s(1:500))
title('Plot 12 - multi-tone message signal')

```

```

xlabel('Time')
ylabel('Amplitude')

[specs,Hz]=spectral(s,delta_t); %generate the spectrum

%Plot 13

subplot(212), %plot the spectrum
plot(Hz,specs, 'g')
title('Plot 13 - multi-tone message signal spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

%B. Observe the conventional AM signal and spectrum

convams=conv_am(s,delta_t,fc,m);

%Plot 14

subplot(211), %plot the conventional AM modulated signal
plot(t(1:500),convams(1:500))
title('Plot 14 - multi-tone conventional AM signal')
xlabel('Time')
ylabel('Amplitude')

[convamspec,Hz,fftconvams]=spectral(convams,delta_t); %generate the modulated
                                                    %spectrum

%Plot 17

subplot(212), %plot the modulated spectrum
plot(Hz,convamspec, 'b')
title('Plot 15 - multi-tone conventional AM spectrum')
xlabel('Time')
ylabel('Amplitude')

pause
clg

%C. Recover and detect the conventional AM signal

    %recover and filter
    filtsig=recoverm(fftconvams,'idealbnd',Hz,cutoff1,cutoff2);
    filtsig=filtsig(1:300); %shorten detsig
    envsig=envlope(filtsig); %envelope detection
    bigsig=(envsig-1)/m; %remove DC value, divide by m

```

```

biggersig=bigsig*max_s; %amplify the signal

%plot the complex envelope-detected signal
%Plot 16 %over the message signal

subplot(211),
plot(t(151:250),[filtsig(151:250);envsig(151:250)])
title('Plot 16 - multi-tone filtered and envelope-detected signals')
xlabel('Time')
ylabel('Amplitude')

%Plot 17 %plot the message signal over the amplified signal

subplot(212),
plot(t(151:250),biggersig(151:250))
title('Plot 17 - multi-tone envelope-detected and message signals')
xlabel('Time')
ylabel('Amplitude')
hold on
pause
plot(t(151:250),s(151:250), 'b')
hold off

```

EO 3513 Computer-aided Laboratory 8 Key *Frequency Modulation (FM)*

Question 1: Calculate the following values for the single-tone message signal:

peak power
average power
baseband bandwidth

Answer: peak power = $P_p = \frac{A_p^2}{2} \Rightarrow 15^2 / 2 \Rightarrow 112.5$

average power = $P = A_0^2 + \frac{1}{2} \sum_{N=1}^{\infty} (A_N^2 + B_N^2) \Rightarrow 15^2 / 2 \Rightarrow 112.5$

baseband bandwidth = 50 Hz

Question 2: Predict the following values for the single-tone FM signal:

peak power
average power
maximum frequency deviation Δf
transmission bandwidth

Answer: peak power = $P_p = \frac{A_p^2}{2} \Rightarrow 15^2 / 2 \Rightarrow 112.5$

average power = $A^2 / 2 \Rightarrow 15^2 / 2 \Rightarrow 112.5$

maximum frequency deviation = $\beta f_m \Rightarrow 10 * 50 \Rightarrow 500$ Hz

transmission bandwidth = $2 \beta f_m \Rightarrow 2 * 10 * 50 \Rightarrow 1000$ Hz

Question 3: What is the distance between the sidebands in the FM spectrum shown in Plot 4?

Answer: 50 Hz (the value of f_m)

Question 4: From Plot 5, obtain the values representing peak and average power.

Do your theoretical calculations for bandwidth and power agree with the computer-generated values?

Answer: peak power = 112.5
average power = 112.5

Yes - calculations agree.

Question 5: Consult a table of values for Bessel functions (or use the MATLAB "bessel" function). Calculate the amplitude for the spectral components shown in the FM spectrum in Plot 4 for $n = 0$ through 6. List each frequency by its Hz value and sideband number n . Values should be consistent with the amplitudes shown for power spectral density in Plot 5.

Answer:	$n = 0$ (1000 Hz)	$0.2459 * 15 = 3.6885$
	$n = 1$ (950, 1050 Hz)	$0.0435 * 15 = 0.6525$
	$n = 2$ (900, 1100 Hz)	$0.2546 * 15 = 3.819$
	$n = 3$ (850, 1150 Hz)	$0.0584 * 15 = 0.876$
	$n = 4$ (800, 1200 Hz)	$0.2196 * 15 = 3.294$
	$n = 5$ (750, 1250 Hz)	$0.2341 * 15 = 3.5115$
	$n = 6$ (700, 1300 Hz)	$0.0145 * 15 = 0.2175$

Question 6: Calculate the maximum frequency deviation Δf associated with each of the four values of β :

0.1
1
5
20

Answer: $\Delta f = \beta f_m \Rightarrow 0.1 * 50 \Rightarrow 5 \text{ Hz}$
 $\Delta f = \beta f_m \Rightarrow 1 * 50 \Rightarrow 50 \text{ Hz}$
 $\Delta f = \beta f_m \Rightarrow 5 * 50 \Rightarrow 250 \text{ Hz}$
 $\Delta f = \beta f_m \Rightarrow 20 * 50 \Rightarrow 1000 \text{ Hz}$

Question 7: Predict the transmission bandwidth for each of the FM signals referred to in Question 6.

Answer: for $\beta = 0.1$ $B_T \approx 2 f_m \Rightarrow 2 * 50 \Rightarrow 100 \text{ Hz}$
 for $\beta = 1$ $B_T \approx 2 (1 + \beta) f_m \Rightarrow 2 (1 + 1) 50 \Rightarrow 200 \text{ Hz}$
 for $\beta = 5$ $B_T \approx 2 (1 + \beta) f_m \Rightarrow 2 (1 + 5) 50 \Rightarrow 300 \text{ Hz}$
 for $\beta = 20$ $B_T \approx 2 \beta f_m \Rightarrow 2 * 20 * 50 \Rightarrow 2000 \text{ Hz}$

Question 8: Calculate the following values for the multi-tone message signal:

peak power
 average power
 baseband bandwidth

Answer: peak power = $P_p = \frac{A_p^2}{2} \Rightarrow 20^2 / 2 \Rightarrow 200$
 average power = $P = A_0^2 + \frac{1}{2} \sum_{N=1}^{\infty} (A_N^2 + B_N^2) \Rightarrow 8^2 / 2 + 12^2 / 2 \Rightarrow 104$
 baseband bandwidth = 75 Hz

Question 9: Predict the following values for the multi-tone FM signal:

peak power
 average power
 maximum frequency deviation Δf
 transmission bandwidth

Answer: peak power = $P_p = \frac{A_p^2}{2} \Rightarrow 20^2 / 2 \Rightarrow 200$
 average power = $A^2 / 2 \Rightarrow 20^2 / 2 \Rightarrow 200$
 maximum frequency deviation = $\beta f_m \Rightarrow 10 * 75 \Rightarrow 750 \text{ Hz}$
 transmission bandwidth = $2 \beta f_m \Rightarrow 2 * 10 * 75 \Rightarrow 1500 \text{ Hz}$

Question 10: What is the distance between the sidebands in the FM spectrum shown in Plot 13?

Answer: 50 Hz (the value of f_m)

Question 11: From Plot 14, obtain the values representing peak and average power.

Do your theoretical calculations for bandwidth and power agree with the computer-generated values?

Answer: peak power = 200
average power = 200

Yes - calculations agree.

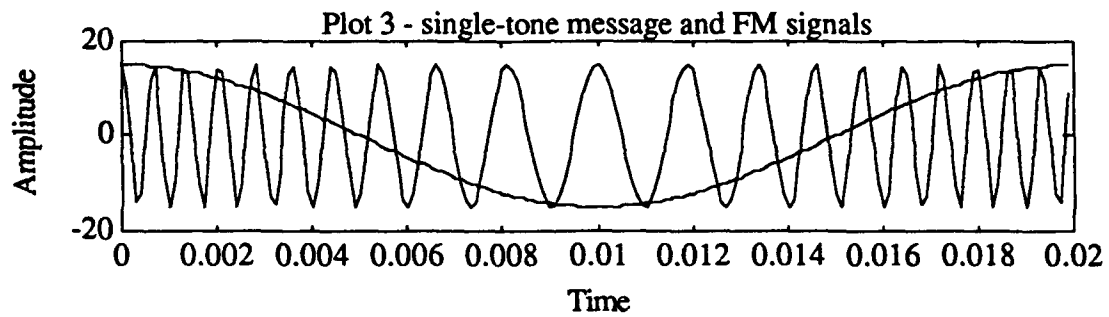
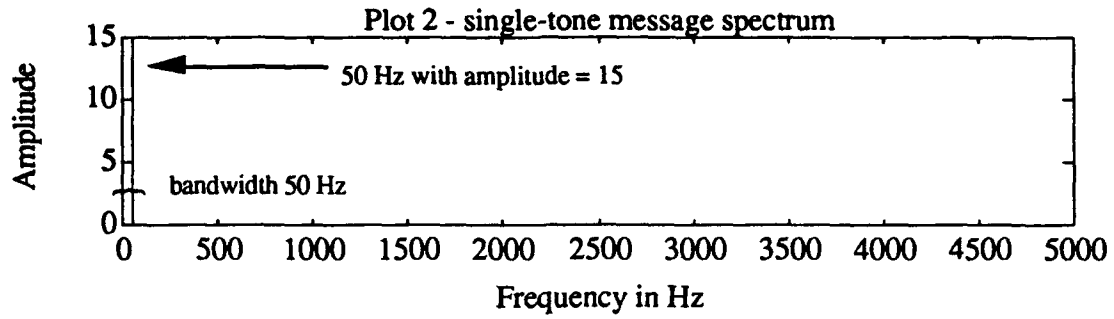
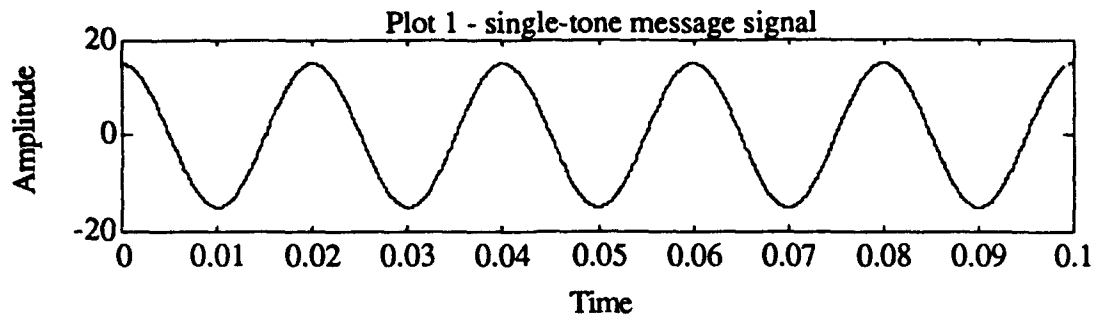
Question 12: Calculate the value of β associated with each of the four values of Δf :

25
100
500
1000

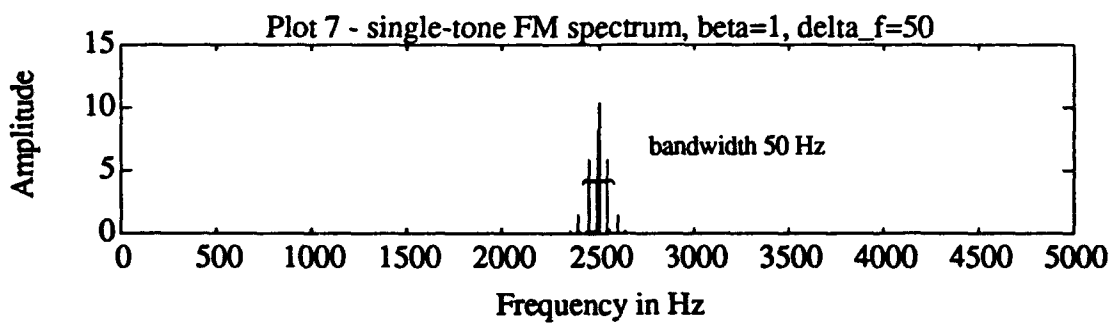
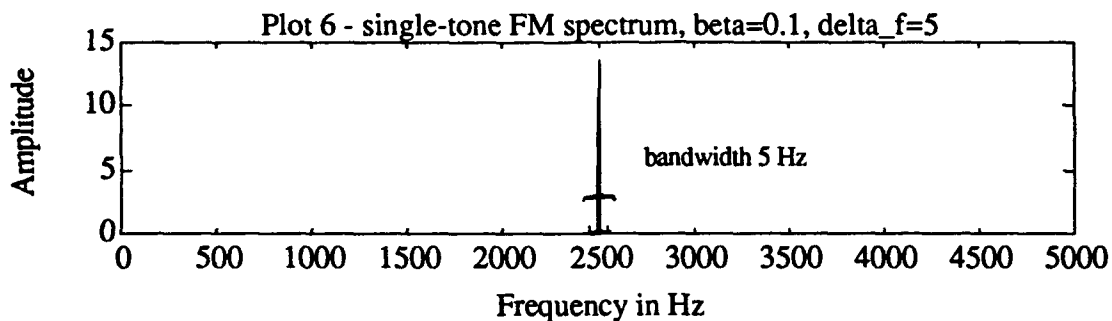
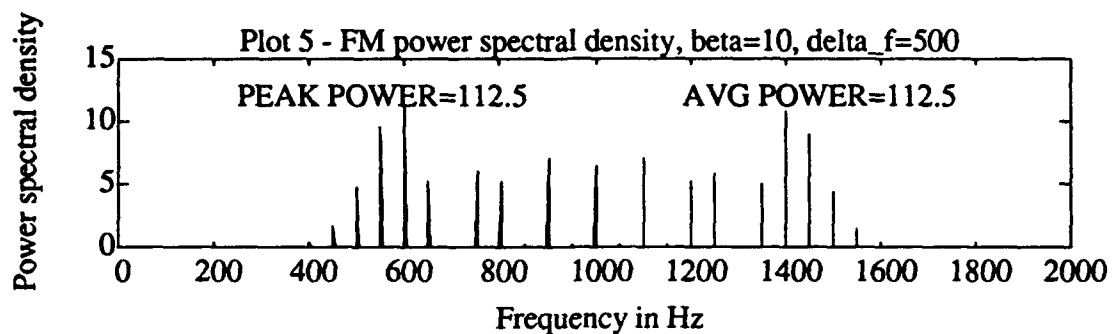
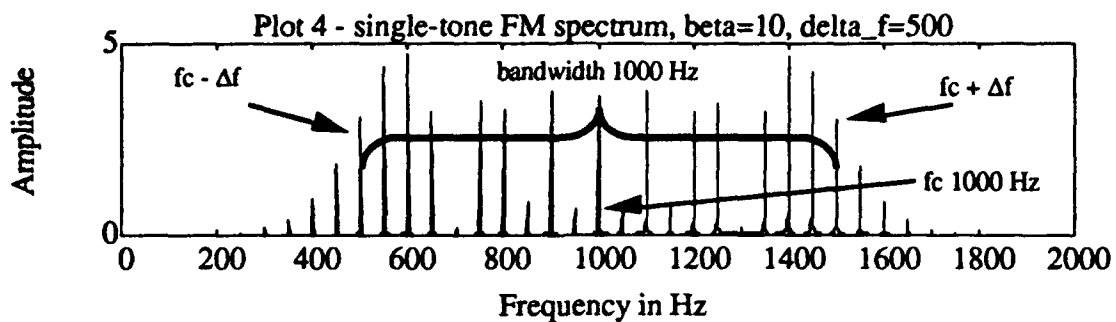
Answer: $\beta = \Delta f / f_m \Rightarrow 25 / 75 \Rightarrow 0.33$
 $\beta = \Delta f / f_m \Rightarrow 100 / 75 \Rightarrow 1.33$
 $\beta = \Delta f / f_m \Rightarrow 500 / 75 \Rightarrow 6.67$
 $\beta = \Delta f / f_m \Rightarrow 1000 / 75 \Rightarrow 13.33$

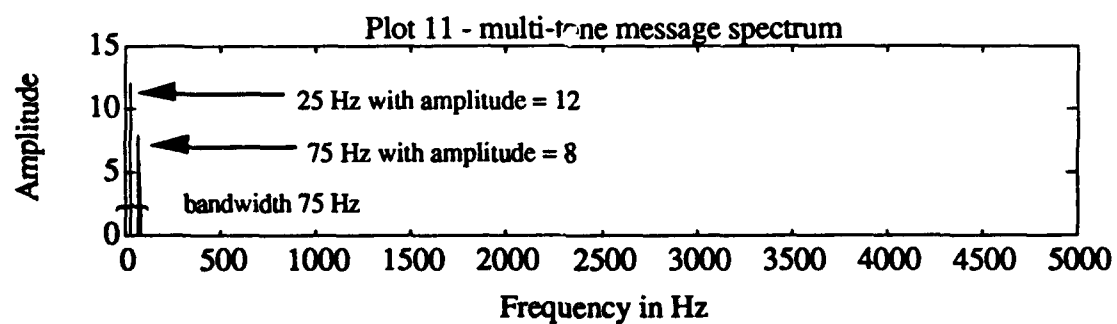
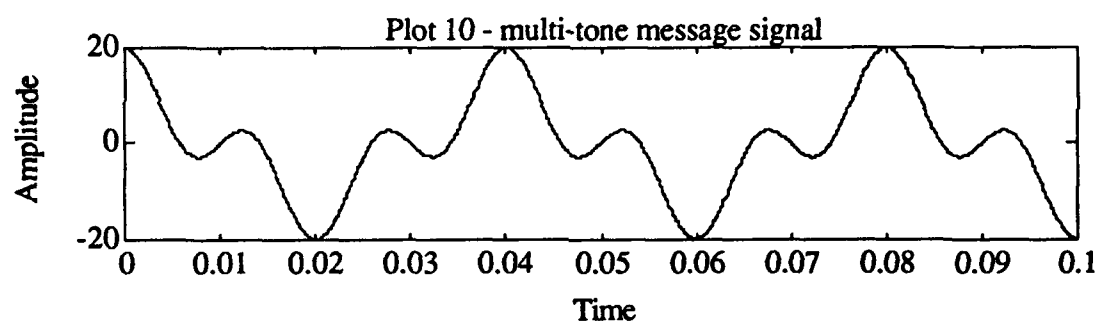
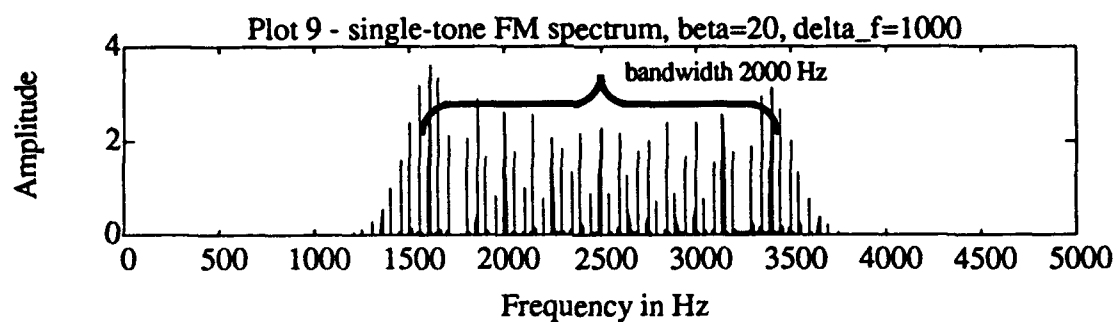
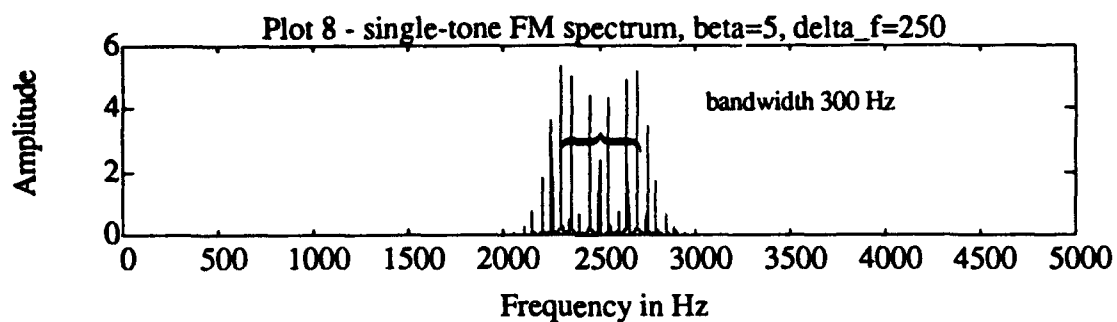
Question 13: Predict the transmission bandwidth for each of the FM signals referred to in Question 12.

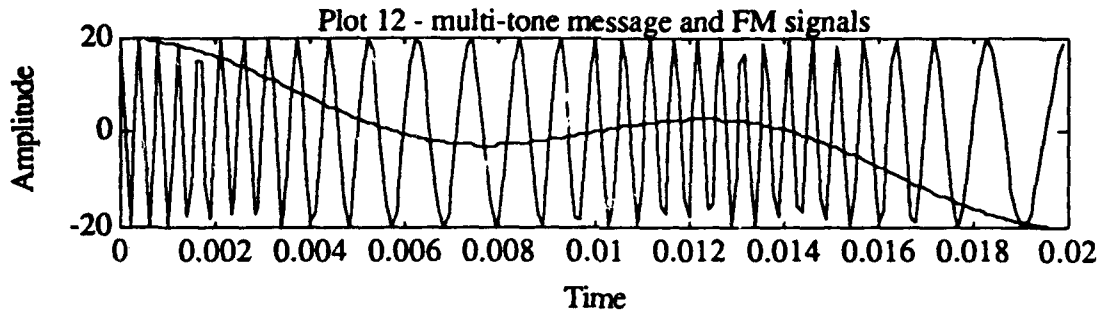
Answer: for $\beta = 0.33$ $B_T \approx 2 (1 + \beta) f_m \Rightarrow 2 (1 + 0.33) 75 \Rightarrow 250 \text{ Hz}$
for $\beta = 1.33$ $B_T \approx 2 (1 + \beta) f_m \Rightarrow 2 (1 + 1.33) 75 \Rightarrow 325 \text{ Hz}$
for $\beta = 6.67$ $B_T \approx 2 (1 + \beta) f_m \Rightarrow 2 (1 + 6.67) 75 \Rightarrow 725 \text{ Hz}$
for $\beta = 13.33$ $B_T \approx 2 \beta f_m \Rightarrow 2 * 13.33 * 75 \Rightarrow 2000 \text{ Hz}$



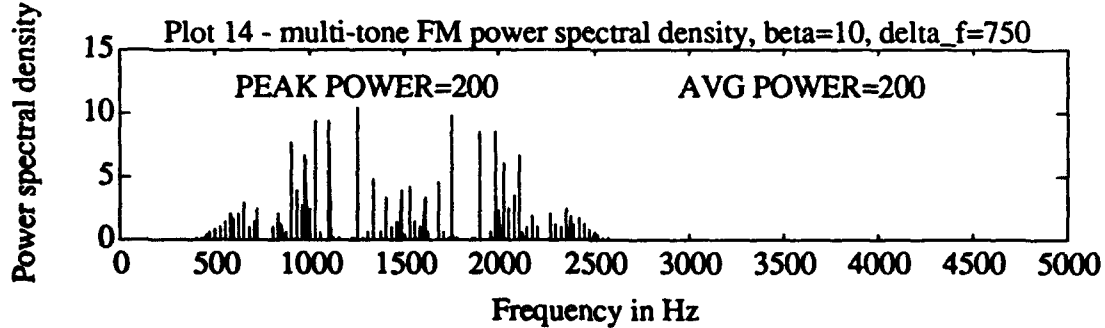
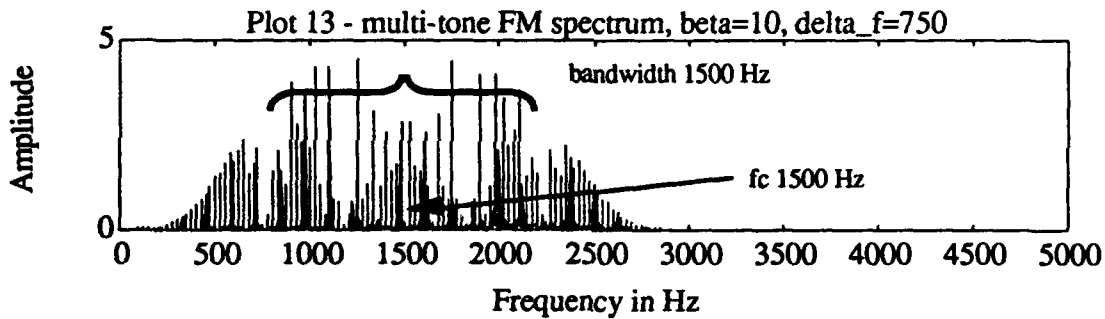
NO PLOT HERE--JUST PRESS RETURN

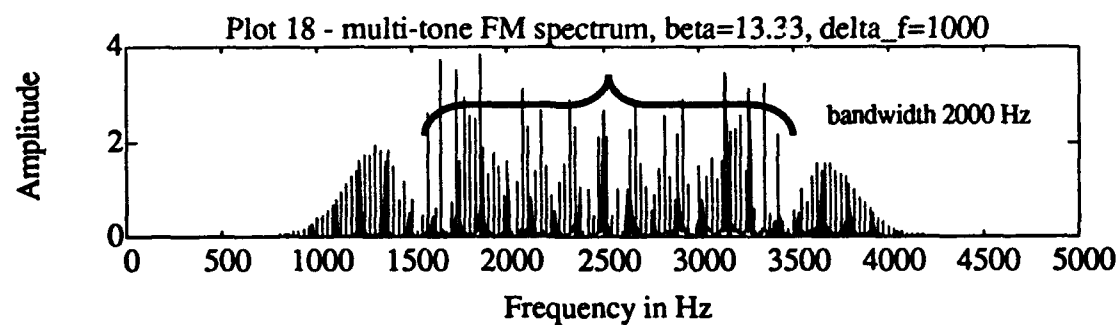
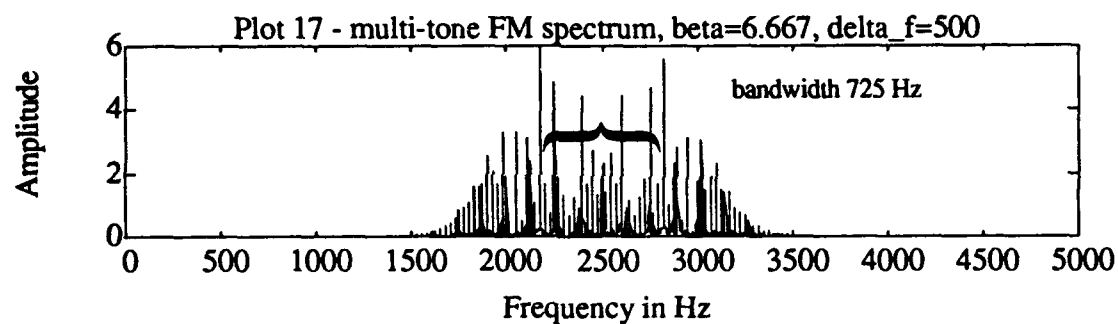
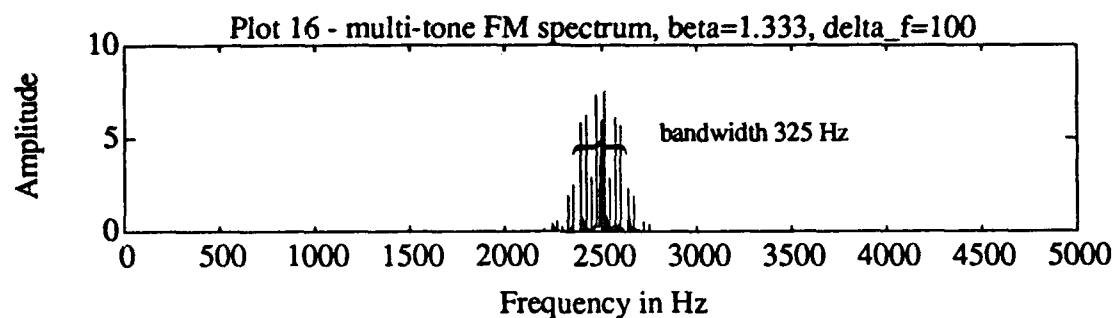
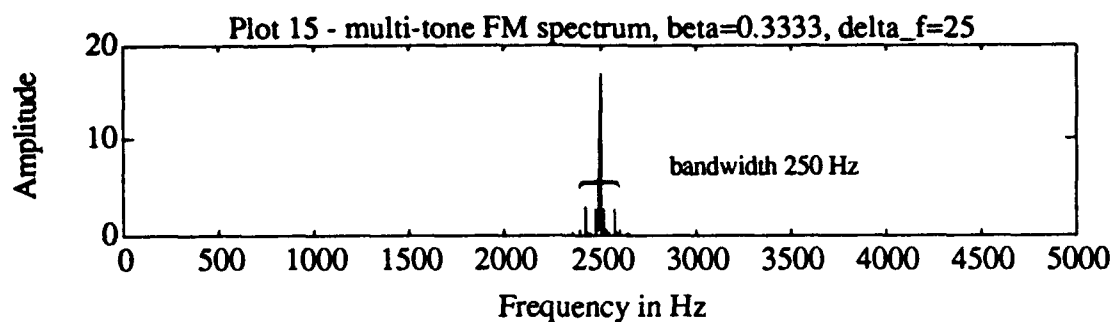






NO PLOT HERE--JUST PRESS RETURN





lab8scr.m

%Computer-aided Lab 8 script for student use

%%

%Computer-aided Lab 8 Frequency Modulation (FM)

%%

%Part 1--Observe the FM modulation process for single-tone input

%A. Calculate theoretical average power, peak power, and bandwidth

% for the single-tone message signal

clear

clg

delta_t=.0001;

t=0:delta_t:1;

Ac=15; %FM signal amplitude for single-tone message

fc=1000; %FM signal frequency for single-tone

theta=0; %single-tone value

fm=50; %single-tone message signal frequency

s=15*cos(2*pi*fm*t); %single-tone signal

%B. Observe the single-tone message signal and its spectrum

%Plot 1

subplot(211),

plot(t(1:1000),s(1:1000))

title('Plot 1 - single-tone message signal')

xlabel('Time')

ylabel('Amplitude')

[msg_spec,HZ]=spectral(s,delta_t);

%Plot 2

subplot(212),

plot(HZ,msg_spec)

title('Plot 2 - single-tone message spectrum')

xlabel('Frequency in Hz')

ylabel('Amplitude')

pause

clg

%C. Observe the process of FM modulation

Computer-aided Laboratory 8 Key—page 10

```

beta=10;
                                %generate the FM signal
[fm_sig,delta_f,beta]=fm_mod(t,Ac,fc,fm,theta,'none',beta);

```

```
%Plot 3
```

```

subplot(211),
plot(t(1:200),[s(1:200);fm_sig(1:200)])
title('Plot 3 - single-tone message and FM signals')
xlabel('Time')
ylabel('Amplitude')

```

```

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

```

```

pause
clg

```

```
%D. Observe the spectrum of the FM signal
```

```
[fm_spec,Hz]=spectral(fm_sig,delta_t);
```

```
%Plot 4
```

```

subplot(211),
plot(Hz(1:2000),fm_spec(1:2000))
title(['Plot 4 - single-tone FM spectrum, beta=',...
      num2str(beta) ', delta_f=' num2str(delta_f)])
xlabel('Frequency in Hz')
ylabel('Amplitude')

```

```
%E. Verify the power and bandwidth of the FM signal
```

```
psdfm=psd(fm_sig,delta_t);
```

```

fm_pk_pwr_sngl=(max(fm_sig)^2)/2;
fm_avg_pwr_sngl=sum(psdfm);
str1=fm_pk_pwr_sngl;
str2=fm_avg_pwr_sngl;

```

```
%Plot 5
```

```

subplot(212),
plot(Hz(1:2000),psdfm(1:2000))
title(['Plot 5 - FM power spectral density, beta=...
      num2str(beta) ', delta_f=' num2str(delta_f)])
xlabel('Frequency in Hz')

```



```

ylabel('Power spectral density')
text(.2,.35,['PEAK POWER=' num2str(str1)],'sc')
text(.6,.35,['AVG POWER=' num2str(str2)],'sc')

pause
clg

%F. Control the bandwidth of the FM signal by varying beta
    %Fix beta at values of 0.1, 1, 5, and 20
    %Modulate at 2500 Hz to center the spectrum
fc=2500;
beta=0.1;

[fm_sig,delta_f,beta]=fm_mod(t,Ac,fc,fm,theta,'none',beta);
fm_spec=spectral(fm_sig,delta_t);

%Plot 6

subplot(211),
plot(Hz,fm_spec)
title(['Plot 6 - single-tone FM spectrum, beta=' num2str(beta)...
    ', delta_f=' num2str(delta_f)])
xlabel('Frequency in Hz')
ylabel('Amplitude')

beta=1;

[fm_sig,delta_f,beta]=fm_mod(t,Ac,fc,fm,theta,'none',beta);
fm_spec=spectral(fm_sig,delta_t);

%Plot 7

subplot(212),
plot(Hz,fm_spec)
title(['Plot 7 - single-tone FM spectrum, beta=' num2str(beta)...
    ', delta_f=' num2str(delta_f)])
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

beta=5;

[fm_sig,delta_f,beta]=fm_mod(t,Ac,fc,fm,theta,'none',beta);
fm_spec=spectral(fm_sig,delta_t);

%Plot 8

```

```

subplot(211),
plot(Hz, fm_spec)
title(['Plot 8 - single-tone FM spectrum, beta=' num2str(beta)...
      ', delta_f=' num2str(delta_f)])
xlabel('Frequency in Hz')
ylabel('Amplitude')

beta=20;

[fm_sig,delta_f,beta]=fm_mod(t,Ac,fc,fm,theta,'none',beta);
fm_spec=spectral(fm_sig,delta_t);

%Plot 9

subplot(212),
plot(Hz, fm_spec)
title(['Plot 9 - single-tone FM spectrum, beta=' num2str(beta)...
      ', delta_f=' num2str(delta_f)])
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Part 2--Observe the FM modulation process for multi-tone input
%A. Calculate theoretical average power, peak power, and bandwidth
%   for the multi-tone message signal

clear
clg

delta_t=.0001; %set signal and modulation variables
t=0:delta_t:1;

Ac=20; %FM signal amplitude for multi-tone message
fc=1500; %FM signal frequency for multi-tone
theta=[0 0]; %multi-tone value
fm=[75 25]; %multi-tone frequency vector
s=8*cos(2*pi*75*t)+12*cos(2*pi*25*t); %multi-tone signal

%B. Observe the multi-tone message signal and its spectrum

%Plot 10

subplot(211),
plot(t(1:1000),s(1:1000))

```

```

title('Plot 10 - multi-tone message signal')
xlabel('Time')
ylabel('Amplitude')

[msg_spec,Hz]=spectral(s,delta_t);

%Plot 11

subplot(212),
plot(Hz,msg_spec)
title('Plot 11 - multi-tone message spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

%C. Observe the process of FM modulation

beta=10;
                                %generate the fm signal
[fm_sig,delta_f,beta]=fm_mod(t,Ac,fc,fm,theta,'none',beta);

%Plot 12

subplot(211),
plot(t(1:200),[s(1:200);fm_sig(1:200)])
title('Plot 12 - multi-tone message and FM signals')
xlabel('Time')
ylabel('Amplitude')

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

pause
clg

%D. Observe the spectrum of the FM signal

[fm_spec,Hz]=spectral(fm_sig,delta_t);

%Plot 13

subplot(211),
plot(Hz,fm_spec)
title(['Plot 13 - multi-tone FM spectrum, beta=',...
      num2str(beta) ' , delta_f=' num2str(delta_f)])
xlabel('Frequency in Hz')

```

```

ylabel('Amplitude')

%E. Verify the power and bandwidth of the FM signal

psdfm=psd(fm_sig,delta_t);

fm_pk_pwr_mlt=(max(fm_sig)^2)/2;
fm_avg_pwr_mlt=sum(psdfm);
str1=fm_pk_pwr_mlt;
str2=fm_avg_pwr_mlt;

%Plot 14

subplot(212),
plot(Hz,psdfm)
title(['Plot 14 - multi-tone FM power spectral density, beta='...
      num2str(beta) ', delta_f=' num2str(delta_f)])
xlabel('Frequency in Hz')
ylabel('Power spectral density')
text(.2,.35,['PEAK POWER=' num2str(str1)],'sc')
text(.6,.35,['AVG POWER=' num2str(str2)],'sc')

pause
clg

%F. Control the bandwidth of the FM signal by varying delta_f
      %Fix delta_f at values of 25, 100, 500, and 1000
      %Modulate at 2500 Hz to center the spectrum

fc=2500;
delta_f=25;

[fm_sig,delta_f,beta]=fm_mod(t,Ac,fc,fm,theta,delta_f,'none');
fm_spec=spectral(fm_sig,delta_t);

%Plot 15

subplot(211),
plot(Hz,fm_spec)
title(['Plot 15 - multi-tone FM spectrum, beta=' num2str(beta)...
      ', delta_f=' num2str(delta_f)])
xlabel('Frequency in Hz')
ylabel('Amplitude')

delta_f=100;

[fm_sig,delta_f,beta]=fm_mod(t,Ac,fc,fm,theta,delta_f,'none');
fm_spec=spectral(fm_sig,delta_t);

```

```

%Plot 16

subplot(212),
plot(Hz, fm_spec)
title(['Plot 16 - multi-tone FM spectrum, beta=' num2str(beta)...
      ', delta_f=' num2str(delta_f)])
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause

clg

delta_f=500;

[fm_sig,delta_f,beta]=fm_mod(t,Ac,fc,fm,theta,delta_f,'none');
fm_spec=spectral(fm_sig,delta_t);

%Plot 17

subplot(211),
plot(Hz, fm_spec)
title(['Plot 17 - multi-tone FM spectrum, beta=' num2str(beta)...
      ', delta_f=' num2str(delta_f)])
xlabel('Frequency in Hz')
ylabel('Amplitude')

delta_f=1000;

[fm_sig,delta_f,beta]=fm_mod(t,Ac,fc,fm,theta,delta_f,'none');
fm_spec=spectral(fm_sig,delta_t);

%Plot 18

subplot(212),
plot(Hz, fm_spec)
title(['Plot 18 - multi-tone FM spectrum, beta=' num2str(beta)...
      ', delta_f=' num2str(delta_f)])
xlabel('Frequency in Hz')
ylabel('Amplitude')

```

EO 3513 Computer-aided Laboratory 9 Key
Radio Frequency Digital Modulation Methods
(ASK, FSK, BPSK, and QPSK)

Answers will vary slightly due to the random bitstream generation.

Question 1: Calculate the bit duration τ for this signal.

From the command window, obtain the values of the first 10 bits in the bitstream. Record these values.

Answer: bit duration = $1/\text{bit rate} \Rightarrow 1/100 \Rightarrow 0.01$ seconds

ask_bits = 0 1 0 1 1 0 1 1 0 0

Question 2: Calculate the approximate baseband bandwidth of the NRZL unipolar digital message signal.

Answer: baseband bandwidth = $0.5/\tau \Rightarrow 0.5/0.01 \Rightarrow 50$ Hz

Question 3: Why is ASK modulation often referred to as “on-off keying”?

Answer: The carrier is turned “on” and “off” to represent the 1’s and 0’s in the digital message signal.

Question 4: Describe a noncoherent method of detection for this ASK signal. Why will this method work for ASK?

Answer: Envelope detection is appropriate for an ASK signal since the only the presence or absence of the signal must be detected. (ASK is a DSB-SC signal.)

Question 5: From the command window, obtain the values of the first 10 bits in the bitstream. Record these values.

Answer: fsk_bits = 0 1 0 1 0 1 1 1 1 1

Question 6: From the command window, obtain the values of the first 10 bits in the bitstream. Record these values.

Answer: bpsk_bits = 1 0 1 1 0 1 1 1 1 1

Question 7: What are the effects in the frequency domain of squaring the BPSK signal?

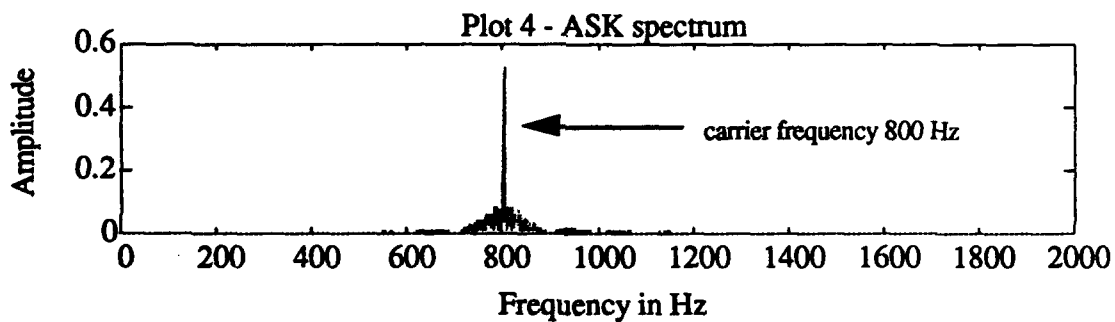
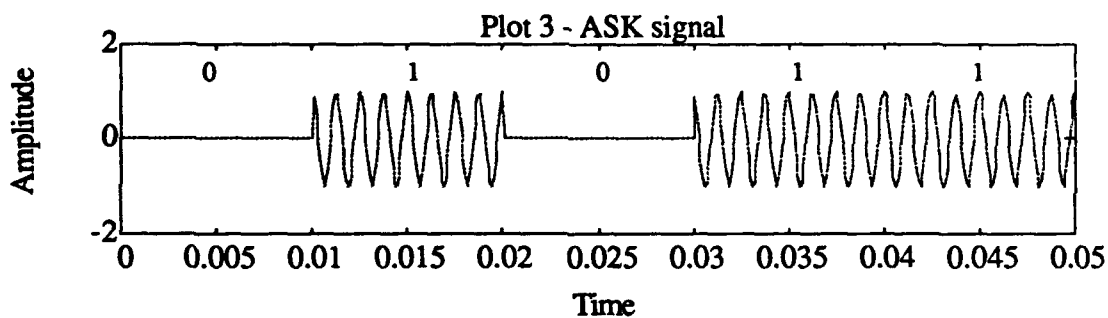
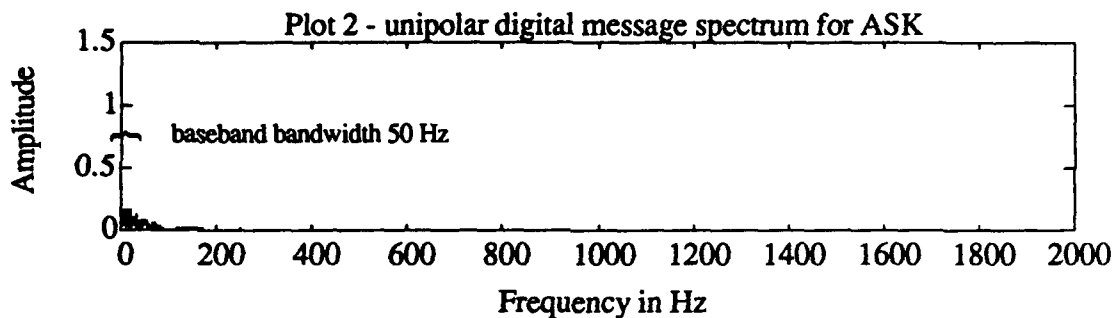
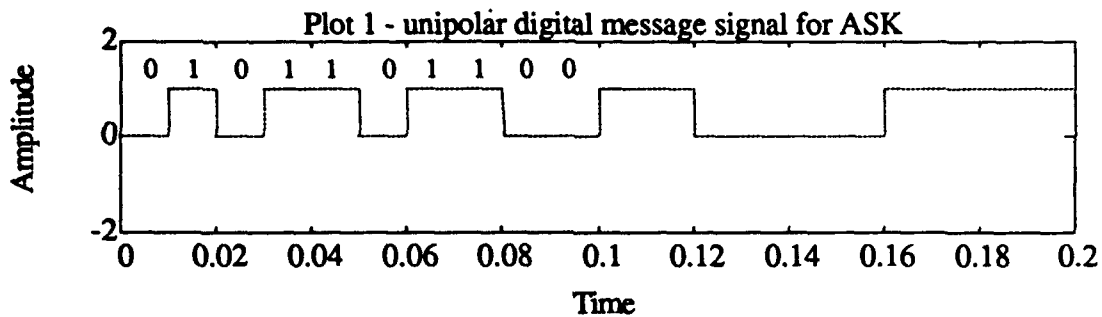
Answer: The spectrum of the BPSK signal is considerably narrowed, and is shifted to a Hz value twice that of the carrier frequency.

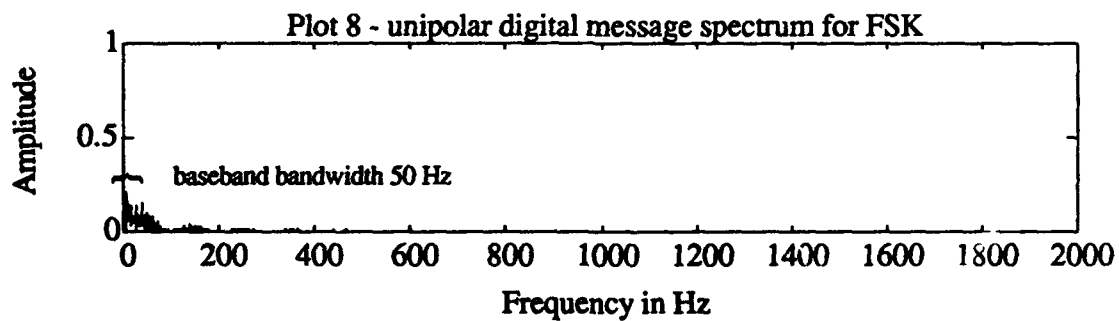
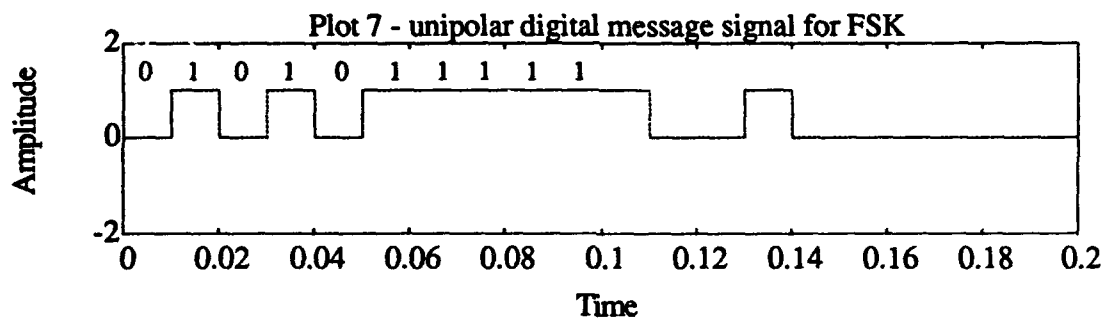
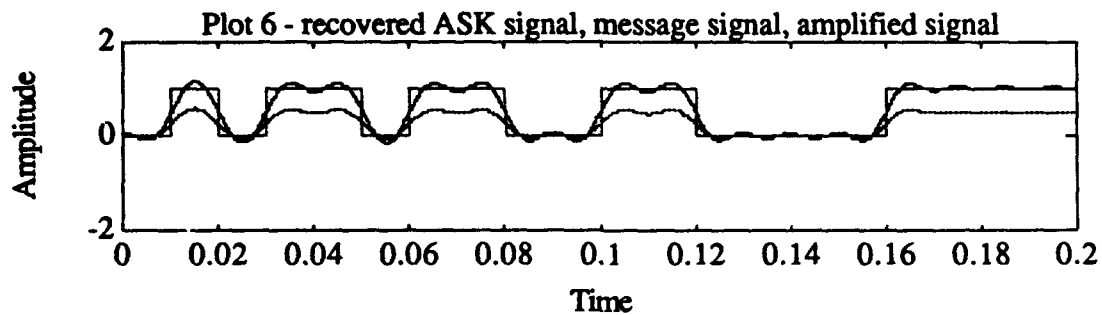
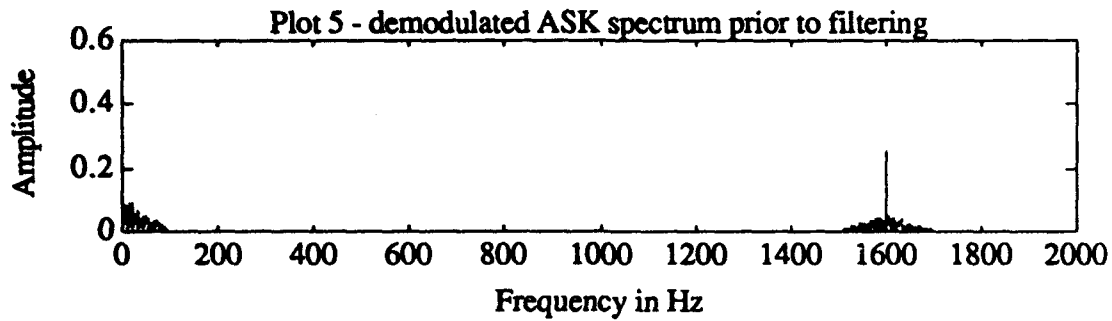
Question 8: From the command window, obtain the values of the first 10 bits in the bitstream. Record these values.

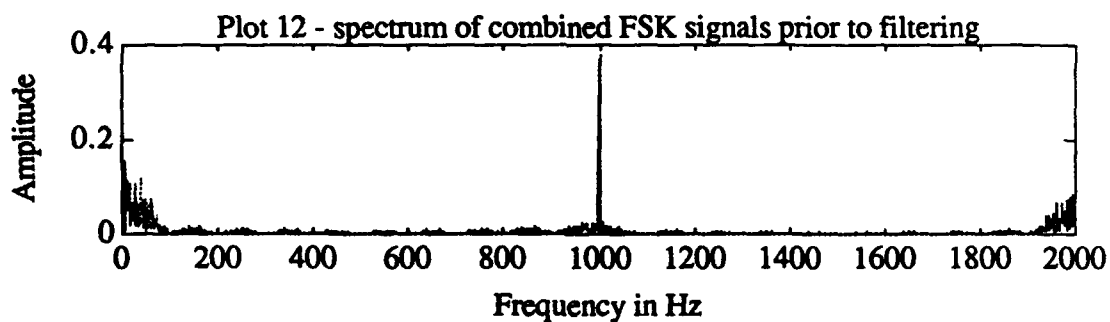
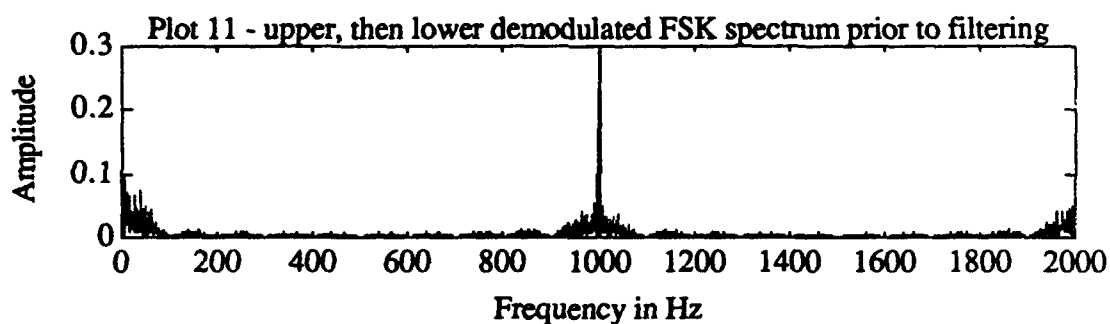
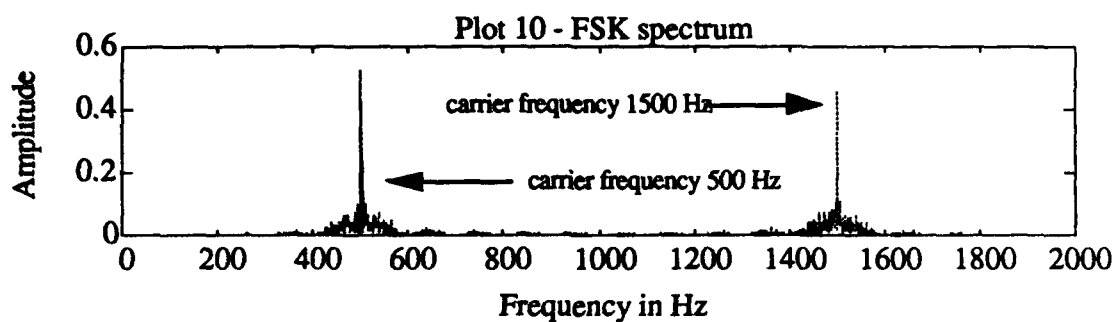
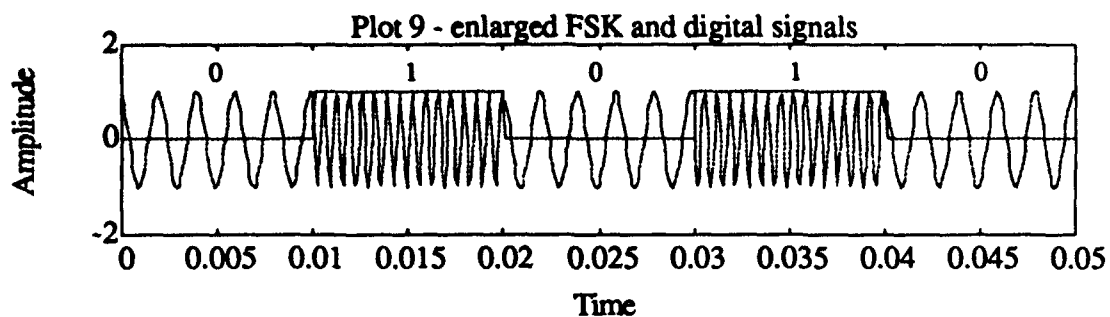
Answer: qpsk_bits = 0 0 0 1 1 0 1 1 0 0

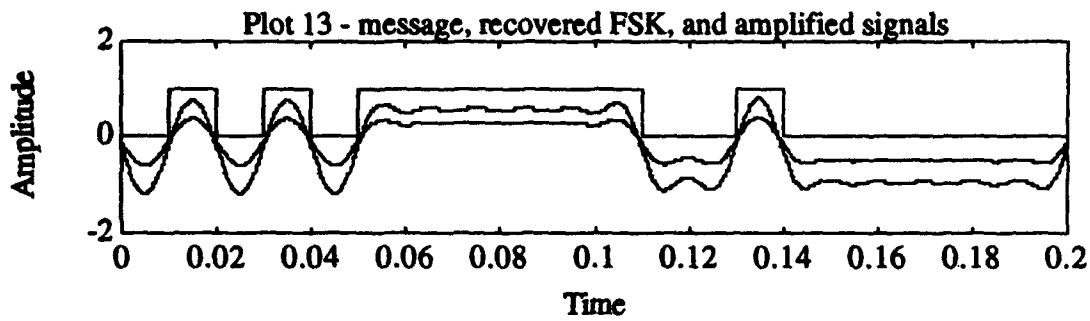
Question 9: What is the chief advantage of quadriphase shift keying over bipolar phase shift keying?

Answer: The information rate of a QPSK signal is twice that of a BPSK signal, with no increase in bandwidth requirements.

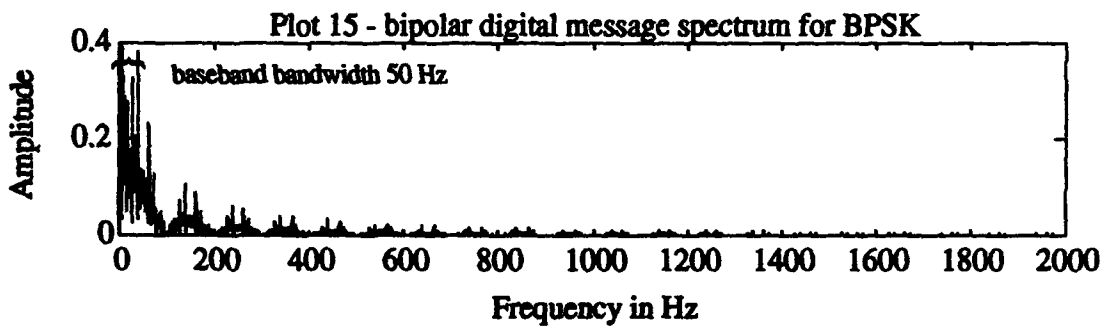
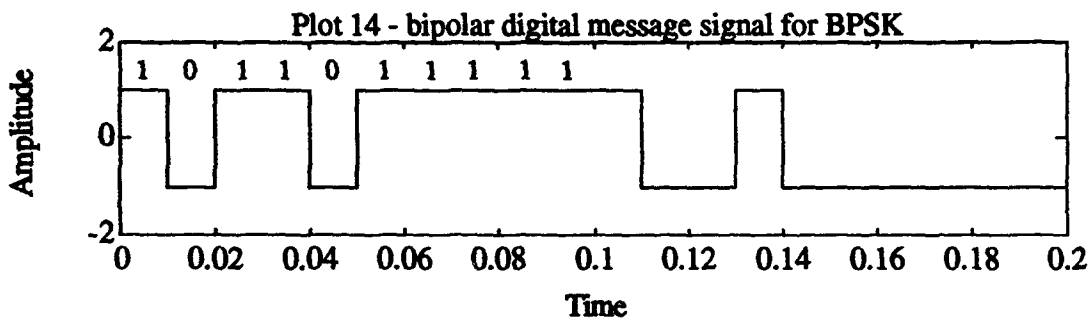


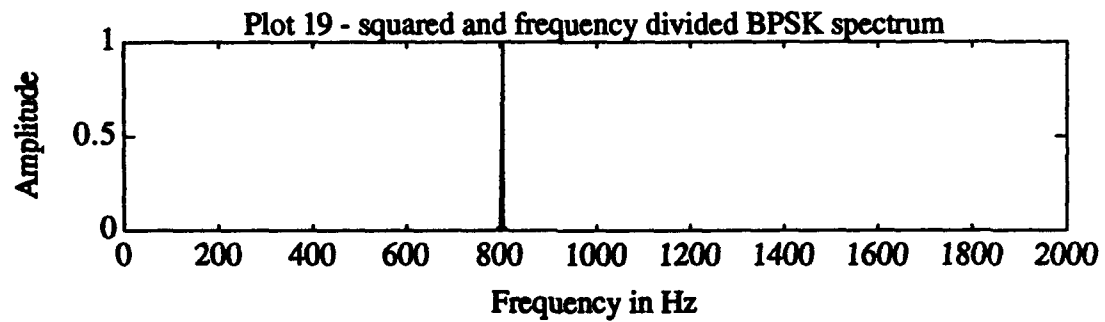
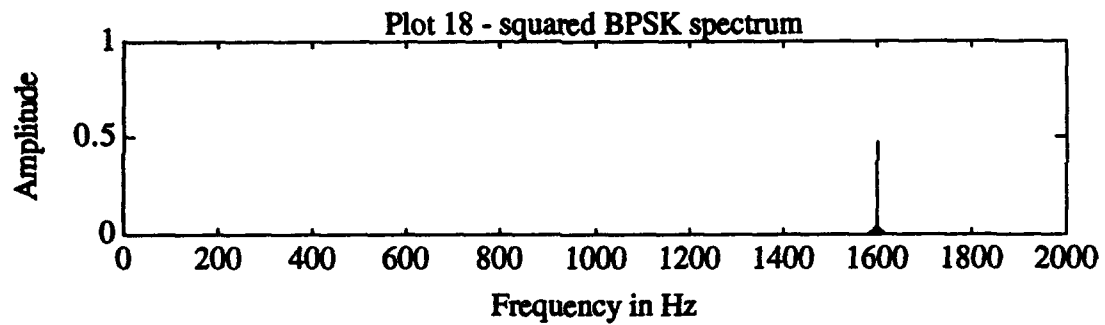
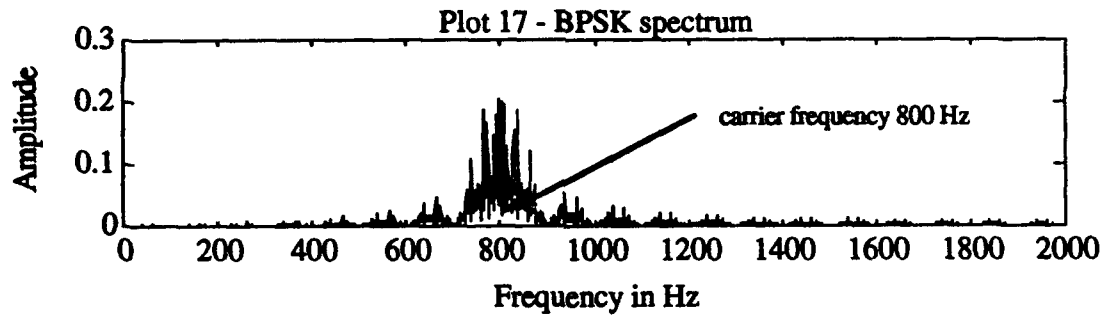
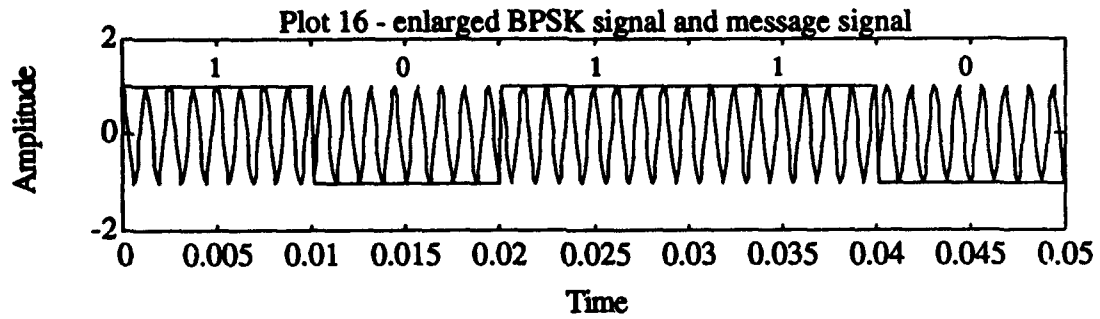


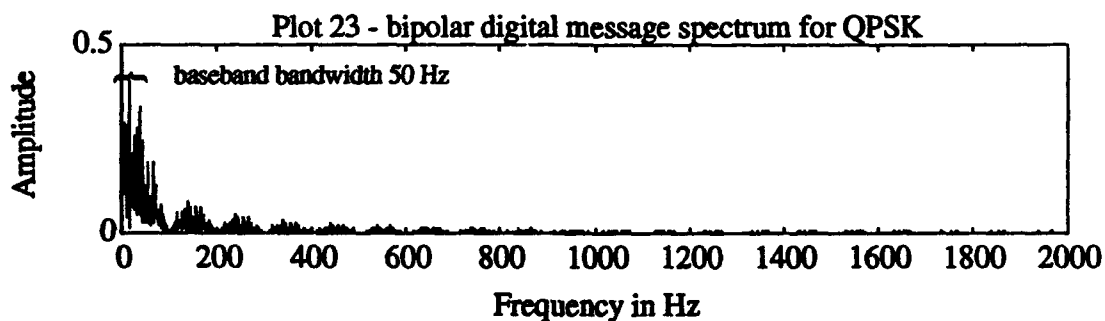
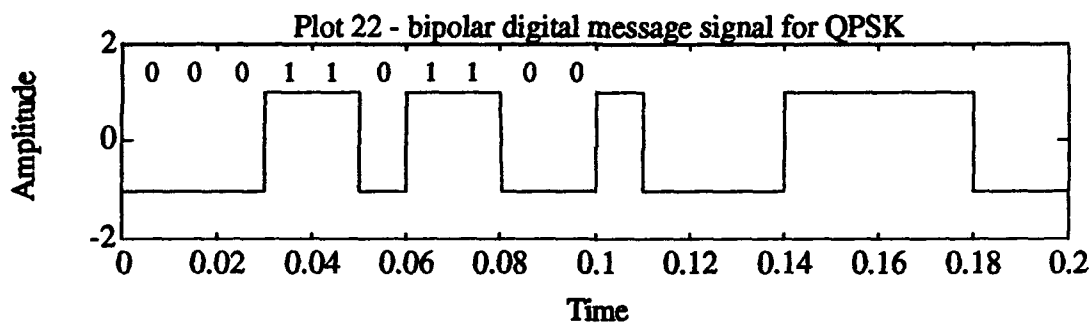
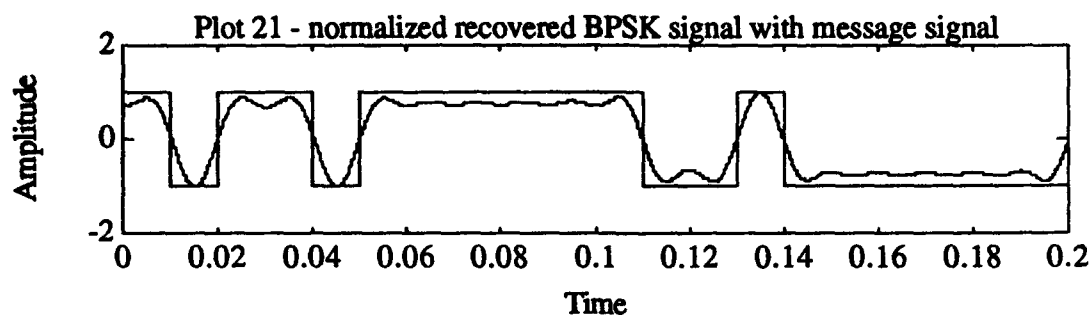
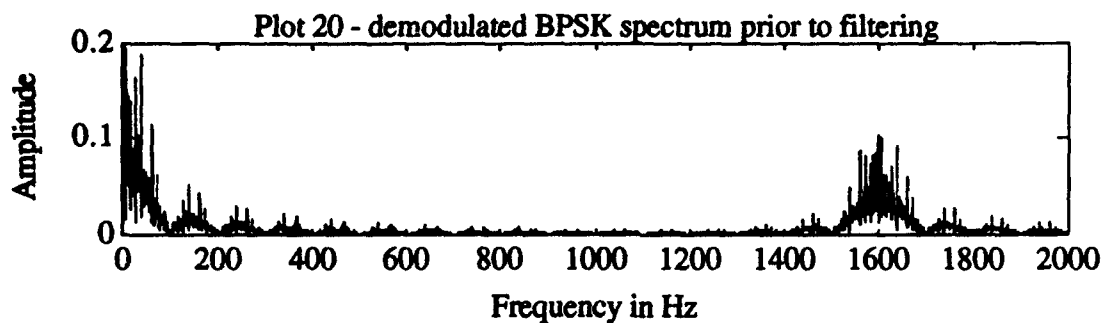


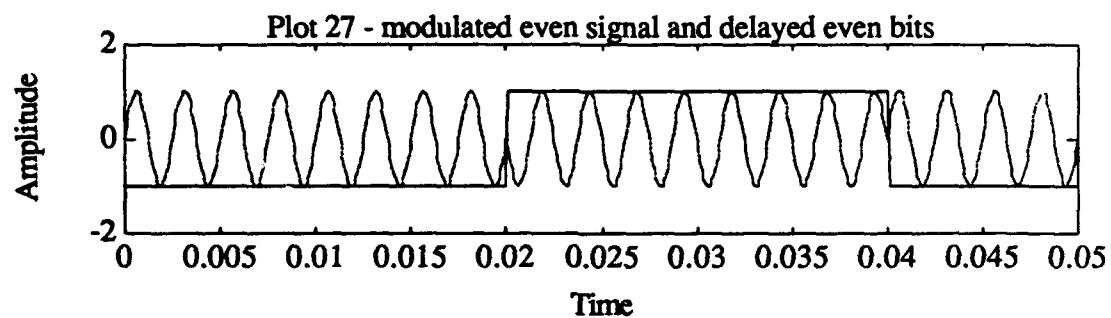
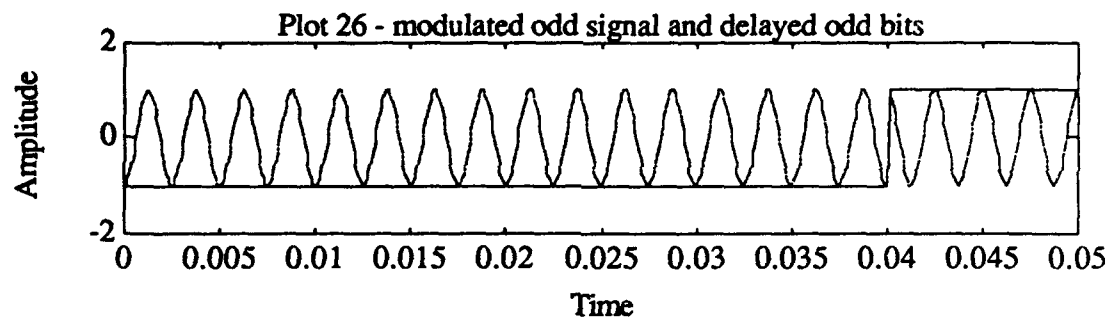
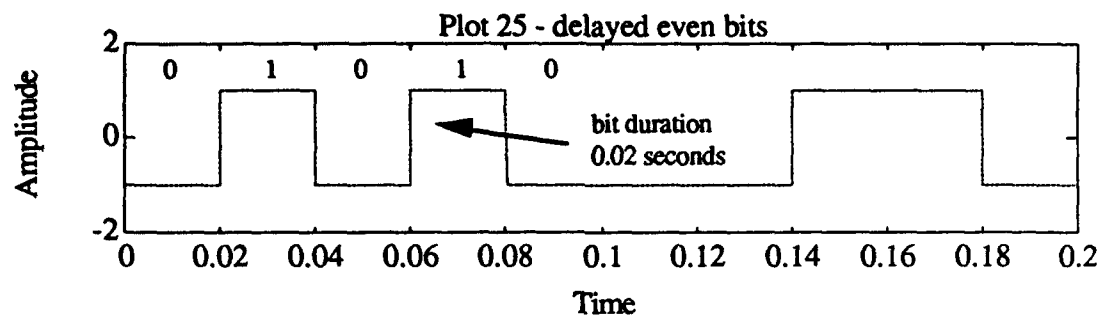
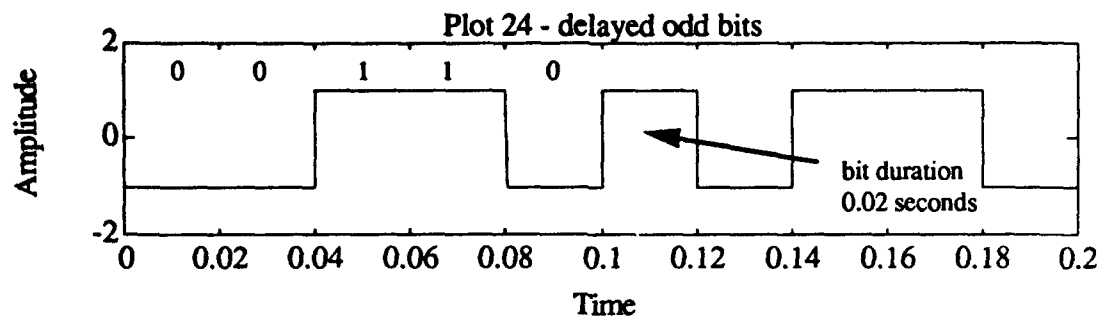


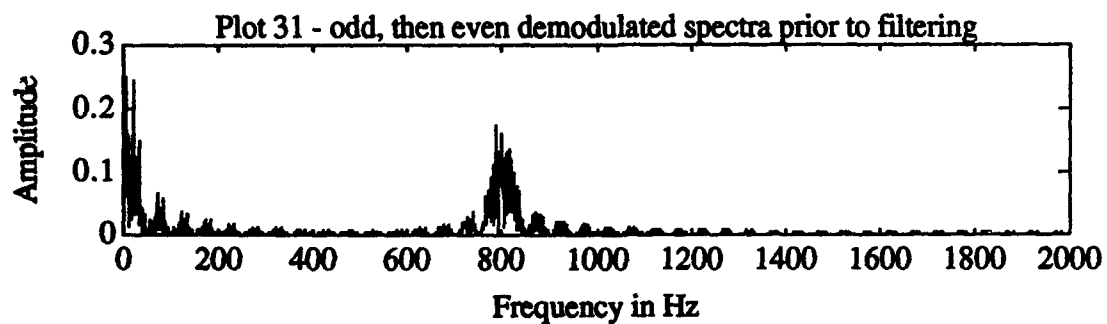
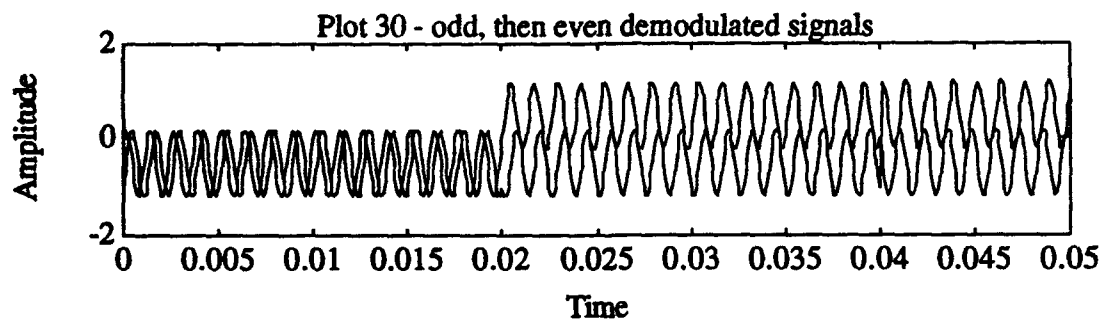
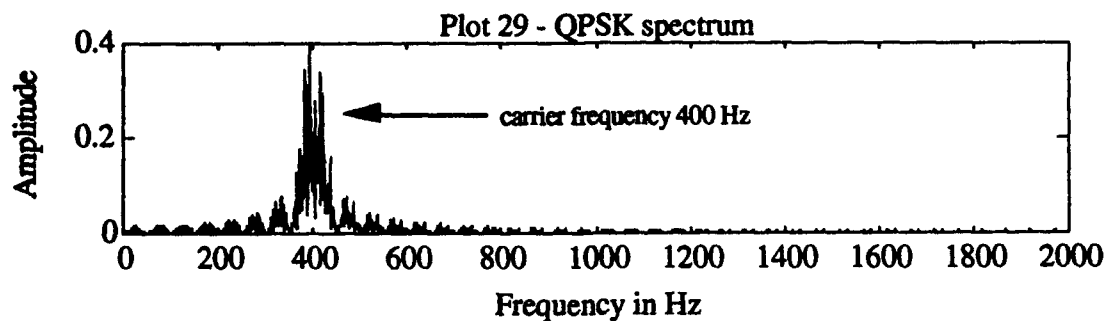
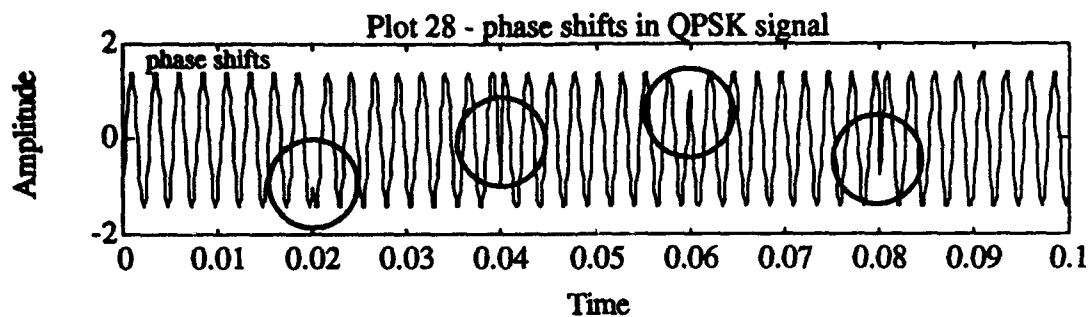
NO PLOT HERE--JUST PRESS RETURN

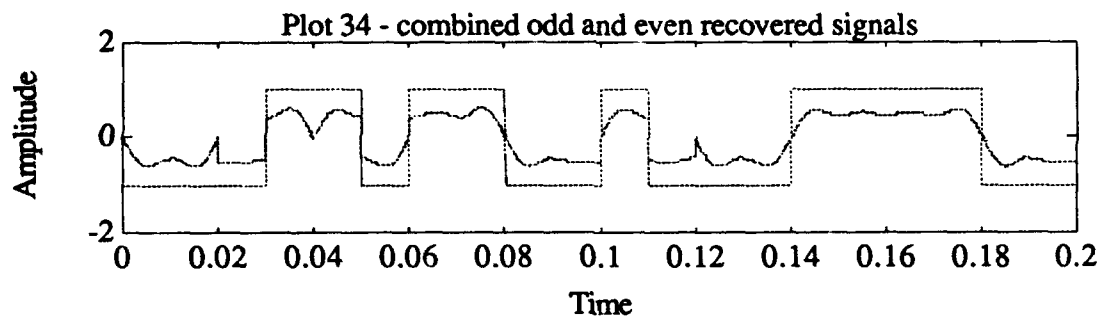
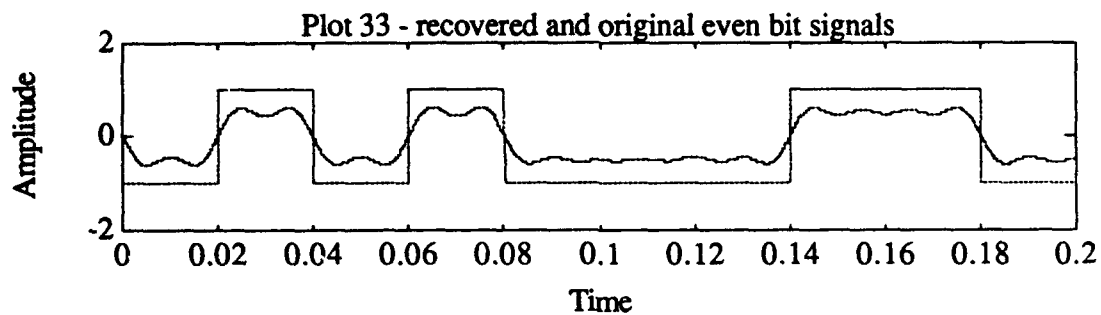
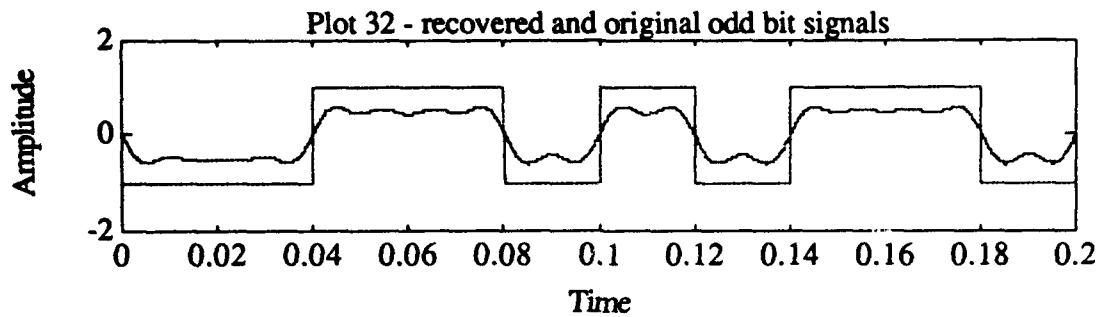












NO PLOT HERE

lab9scr.m

%Lab 9 script for student use

%%

%Computer-aided Laboratory 9 Radio Frequency (RF)

% Digital Modulation

%%

%PART 1--Amplitude Shift Keying (ASK)

%A. Generating the digital message signal

clear

clg

delta_t=.0001;

fc=800;

bitrate=100;

bitstream=round(rand(1:100)); %generate the random bitstream

bitstream(1:3)=[0 1 0]; %ensure one of each bit type

ask_bits=bitstream(1:10) %print the first 10 bits

pause

[nrzlsig,t]=nrzluni(bitstream,delta_t,bitrate); %generate the digital signal

clear bitstream;

big_axis=[0 .2 -2 2]; %set manual scaling for graphs

small_axis=[0 .05 -2 2];

axis(big_axis);

%Plot 1

subplot(211), %plot the digital message signal

plot(t,nrzlsig)

title('Plot 1 - unipolar digital message signal for ASK')

xlabel('Time')

ylabel('Amplitude')

axis; %release manual scaling

[nrzlsec,HZ]=spectral(nrzlsig,delta_t); %generate the message spectrum

%Plot 2

subplot(212),

plot(HZ(1:2000),nrzlsec(1:2000), 'b')

title('Plot 2 - unipolar digital message spectrum for ASK')

```

xlabel('Frequency in Hz')
ylabel('Amplitude')

clear nrzlspec;

pause
clg

%B. Generating the ASK signal

asksig=nrzlsig.*cos(2*pi*fc*t); %generate the ASK signal

axis(small_axis);

%Plot 3

subplot(211), %plot the ASK signal
plot(t,asksig)
title('Plot 3 - ASK signal')
xlabel('Time')
ylabel('Amplitude')

axis; %release manual plot scaling

[askspec,HZ,askfft]=spectral(asksig,delta_t); %generate the ASK spectrum

%Plot 4

subplot(212),
plot(HZ(1:2000),askspec(1:2000), 'g')
title('Plot 4 - ASK spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

%C. Filtering and recovering the ASK signal
%bandpass filter and recover
recask=recoverm(askfft,'idealbnd',HZ,fc-bitrate,fc+bitrate);
clear askfft;
demodask=recask.*cos(2*pi*fc*t); %multiply by carrier in time domain
clear recask;
[demodspec,HZ,demodfft]=spectral(demodask,delta_t); %observe spectrum
clear demodask;

%Plot 5

```

```

subplot(211),
plot(Hz(1:2000),demodspec(1:2000))
title('Plot 5 - demodulated ASK spectrum prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear demodspec;
%lowpass filter and recover
recdemodask=recoverm(demodfft,'ideallow',Hz,bitrate);
clear demodfft;

ampsig=recdemodask*2; %amplify recovered signal

axis(big_axis);

%Plot 6

subplot(212), %plot message, recovered, and amplified signals
plot(t,[recdemodask;nrzlsig])
title('Plot 6 - recovered ASK signal, message signal, amplified signal')
xlabel('Time')
ylabel('Amplitude')
hold on
pause
plot(t,ampsig,'b')
hold off
pause

%%%%%%%%%%%%%%
%PART 2--Frequency Shift Keying (FSK)
%A. Generating the digital message signal

clear
clg

delta_t=.0001; %set variables

low_freq=500;
hi_freq=1500;

bitrate=100;
bitstream=round(rand(1:100)); %generate the random bitstream
bitstream(1:3)=[0 1 0]; %ensure at least one of each bit type
fsk_bits=bitstream(1:10)
pause

big_axis=[0 .2 -2 2]; %set manual scaling for graphs
small_axis=[0 .05 -2 2];

```

```

[nrzlsig,t]=nrzluni(bitstream,delta_t,bitrate); %generate the digital signal
clear bitstream;

axis(big_axis);

%Plot 7

subplot(211),
plot(t,nrzlsig)
title('Plot 7 - unipolar digital message signal for FSK')
xlabel('Time')
ylabel('Amplitude')

axis; %release manual scaling
%observe the NRZL spectrum
[nrzlspec,HZ]=spectral(nrzlsig,delta_t);

%Plot 8

subplot(212),
plot(HZ(1:2000),nrzlspec(1:2000), 'b')
title('Plot 8 - unipolar digital message spectrum for FSK')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clf

%B. Generating the FSK signal

fsksig=fsk(nrzlsig,delta_t,bitrate,low_freq,hi_freq); %generate the FSK signal

axis(small_axis); %manually scale graph

%Plot 9

subplot(211), %FSK signal plotted over message signal
plot(t,[nrzlsig;fsksig])
title('Plot 9 - enlarged FSK and digital signals')
xlabel('Time')
ylabel('Amplitude')

axis; %release

[fskspec,HZ,fskfft]=spectral(fsksig,delta_t); %observe the FSK spectrum

%Plot 10

```

```

subplot(212),
plot(Hz(1:2000),fskspec(1:2000), 'g')
title('Plot 10 - FSK spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

%C. Coherent detection of the FSK signal
    %bandpass filter and recover the FSK signal
recsig=recoverm(fskfft,'idealbnd',Hz,low_freq-bitrate,hi_freq+bitrate);
clear fskfft;

    %lower frequency

demodfsklower=recsig.*cos(2*pi*low_freq*t); %multiply by carrier in time domain
[demodspec1,HZ]=spectral(demodfsklower,delta_t); %generate spectrum

    %upper frequency

demodfskupper=recsig.*cos(2*pi*hi_freq*t); %multiply by carrier in time domain
[demodspecu,HZ]=spectral(demodfskupper,delta_t); %generate spectrum

%Plot 11

subplot(211), %plot upper and lower demodulated signals
plot(Hz(1:2000),demodspecu(1:2000))
title('Plot 11 - upper, then lower demodulated FSK spectrum prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')
hold on
pause
plot(Hz,demodspec1,'b')
hold off
clear demodspec1;clear demodspecu;

combsig=demodfskupper-demodfsklower; %combine signals in time domain
clear demodfsklower;clear demodfskupper;

[combspec,HZ,combfft]=spectral(combsig,delta_t); %spectrum of combined signals
clear combsig;

```

%Plot 12

```
subplot(212),  
plot(Hz(1:2000),combspec(1:2000))  
title('Plot 12 - spectrum of combined FSK signals prior to filtering')  
xlabel('Frequency in Hz')  
ylabel('Amplitude')  
clear combspec;
```

pause

clg

%lowpass filter and recover

```
recfsk=recoverm(combfft,'ideallow',Hz,bitrate);
```

```
clear combfft;
```

```
amprecfsk=recfsk*2; %amplify signal
```

```
axis(big_axis);
```

%Plot 13

```
subplot(211),  
plot(t,[nrzlsig:recfsk])  
title('Plot 13 - message, recovered FSK, and amplified signals')  
xlabel('Time')  
ylabel('Amplitude')  
hold on  
pause  
plot(t,amprecfsk,'b')  
hold off
```

```
subplot(212),  
title('NO PLOT HERE--JUST PRESS RETURN')  
pause
```

%%
%PART 3--Binary Phase Shift Keying (BPSK)
%A. Generating the digital message signal

clear

clg

```
delta_t=.0001;
```

```
fc=800;
```

```
bitrate=100;
```

```
bitstream=round(rand(1:100)); %generate the random bitstream
```

```
bitstream(1:3)=[1 0 1]; %ensure one of each bit type
```

```
bpsk_bits=bitstream(1:10)
```

```

pause

[nrzlsig,t]=nrzlb(bitstream,delta_t,bitrate); %generate the digital
                                         %message signal

clear bitstream;

big_axis=[0 .2 -2 2]; %set manual scaling for graphs
small_axis=[0 .05 -2 2];

axis(big_axis);

%Plot 14

subplot(211), %plot the message signal
plot(t,nrzlsig)
title('Plot 14 - bipolar digital message signal for BPSK')
xlabel('Time')
ylabel('Amplitude')

axis; %release manual scaling

[nrzlspec,HZ]=spectral(nrzlsig,delta_t); %generate the message spectrum

%Plot 15

subplot(212),
plot(HZ(1:2000),nrzlspec(1:2000), 'g')
title('Plot 15 - bipolar digital message spectrum for BPSK')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear nrzlspec;

pause
clg

%B. Generating the BPSK signal

bpsksig=nrzlsig.*cos(2*pi*fc.*t); %generate the BPSK signal

axis(small_axis); %set manual scaling

%Plot 16

subplot(211), %plot the BPSK signal over the message signal
plot(t,bpsksig)
title('Plot 16 - enlarged BPSK signal and message signal')
xlabel('Time')

```

```

ylabel('Amplitude')
hold on
pause
plot(t,nrzlsig,'b')
hold off

[bpskspec,Hz]=spectral(bpsksig,delta_t);

%Plot 17

subplot(212), %plot the BPSK spectrum
plot(Hz(1:2000),bpskspec(1:2000), 'g')
title('Plot 17 - BPSK spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')
clear bpskspec;

pause
clg

%C. Coherent detection of the BPSK signal

squarebpsk=bpsksig.^2; %square the BPSK signal in time domain

div_sig=freq_div(squarebpsk,t,fc); %divide the frequency in half
squarespec=spectral(squarebpsk,delta_t);
[div_spec,Hz,div_fft]=spectral(div_sig,delta_t);

clear squarebpsk;

%Plot 18

subplot(211), %plot the squared signal spectrum--narrow at 2fc
plot(Hz(1:2000),squarespec(1:2000))
title('Plot 18 - squared BPSK spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear squarespec;

%Plot 19

subplot(212),
plot(Hz(1:2000),div_spec(1:2000), 'b')
title('Plot 19 - squared and frequency divided BPSK spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')
clear div_spec;

```



```

pause
clg

rec_bpsk=recoverm(div_fft,'ideallow',Hz,1000); %recover via lowpass filter
clear div_fft;
demodbpsk=rec_bpsk.*bpsksig; %multiply by the received signal
clear bpsksig;
[demodspec,Hz,demodfft]=spectral(demodbpsk,delta_t); %observe spectrum
clear demodbpsk;

%Plot 20

subplot(211),
plot(Hz(1:2000),demodspec(1:2000))
title('Plot 20 - demodulated BPSK spectrum prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')
clear demodspec;

recdemodbpsk=recoverm(demodfft,'ideallow',Hz,birate); %lowpass filter
clear demodfft;clear Hz;

normbpsk=recdemodbpsk/max(recdemodbpsk); %normalize the signal
clear recdemodbpsk;

axis(big_axis);

%Plot 21

subplot(212), %plot the recovered signal
plot(t,[normbpsk;nrzlsig])
title('Plot 21 - normalized recovered BPSK signal with message signal')
xlabel('Time')
ylabel('Amplitude')

axis;

pause

%%%%%%%%%%%%%%
%PART 4--Quadrphase Shift Keying
%A. Generating the digital message signal

clear
clg

delta_t=.0001;

```

```

fc=400;
bitrate=100;

big_axis=[0 .2 -2 2]; %set axes
med_axis=[0 .1 -2 2];
small_axis=[0 .05 -2 2];

bitstream=round(rand(1:100)); %generate the bitstream
bitstream(1:9)=[0 0 0 1 1 0 1 1 0]; %set values to demonstrate phase shifts
qpsk_bits=bitstream(1:10)
pause

[nrzlsig,t]=nrzlb(bitstream,delta_t,bitrate); %generate the digital signal
clear bitstream;
axis(big_axis); %set manual scaling

%Plot 22

subplot(211), %graph the digital signal
plot(t,nrzlsig)
title('Plot 22 - bipolar digital message signal for QPSK')
xlabel('Time')
ylabel('Amplitude')

axis; %release manual scaling

[nrzlspec,Hz]=spectral(nrzlsig,delta_t); %generate the message spectrum

%Plot 23

subplot(212),
plot(Hz(1:2000),nrzlspec(1:2000))
title('Plot 23 - bipolar digital message spectrum for QPSK')
xlabel('Frequency in Hz')
ylabel('Amplitude')
clear nrzlspec;

pause
clg

%B. Generating the QPSK signal

[nrzloff,nrzleven]=ser_par(nrzlsig,delta_t,bitrate); %put signal through
%serial-to-parallel
%converter

axis(big_axis);

%Plot 24

```

```

subplot(211),    %graph the delayed digital signal--odd bits
plot(t,nrzlodd, 'b')
title('Plot 24 - delayed odd bits')
xlabel('Time')
ylabel('Amplitude')

%Plot 25

subplot(212),    %graph the delayed digital signal--even bits
plot(t,nrzleven, 'g')
title('Plot 25 - delayed even bits')
xlabel('Time')
ylabel('Amplitude')

axis;

pause
clg

cos_mod=nrzlodd.*cos(2*pi*fc.*t); %modulate each signal
sin_mod=nrzleven.*(-sin(2*pi*fc.*t));

axis(small_axis);

%Plot 26

subplot(211),
plot(t,cos_mod)
title('Plot 26 - modulated odd signal and delayed odd bits')
xlabel('Time')
ylabel('Amplitude')
hold on
pause
plot(t,nrzlodd, 'b')
hold off

axis(small_axis);

%Plot 27

subplot(212),
plot(t,sin_mod)
title('Plot 27 - modulated even signal and delayed even bits')
xlabel('Time')
ylabel('Amplitude')
hold on
pause

```

```

plot(t,nrzleven, 'b')
hold off

pause
clg

qpsk_sig=cos_mod+sin_mod; %sum the signals in the time domain
clear cos_mod;clear sin_mod;

%Plot 28

axis(med_axis);

subplot(211),
plot(t,qpsk_sig, 'b')
title('Plot 28 - phase shifts in QPSK signal')
xlabel('Time')
ylabel('Amplitude')

axis;

[qpskspec,HZ]=spectral(qpsk_sig,delta_t); %generate the QPSK spectrum

%Plot 29

subplot(212),
plot(HZ(1:2000),qpskspec(1:2000))
title('Plot 29 - QPSK spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear qpskspec;

pause
clg

%C. Coherent detection of the QPSK signal

upperdemod=qpsk_sig.*cos(2*pi*fc.*t); %demodulate each signal
lowerdemod=qpsk_sig.*(-sin(2*pi*fc.*t)); %using its carrier frequency

clear qpsk_sig;

axis(small_axis);

%Plot 30

subplot(211), %plot upper and lower signals

```

```

plot(t,upperdemod)
title('Plot 30 - odd, then even demodulated signals')
xlabel('Time')
ylabel('Amplitude')
hold on
pause
plot(t,lowerdemod, 'g')
hold off

[upperspec,HZ,upperfft]=spectral(upperdemod,delta_t); %generate spectrum for
clear upperdemod;
[lowerspec,HZ,lowerfft]=spectral(lowerdemod,delta_t); %each signal
clear lowerdemod;

%Plot 31

subplot(212),
plot(HZ(1:2000),upperspec(1:2000))
title('Plot 31 - odd, then even demodulated spectra prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')
hold on
pause
plot(HZ,lowerspec, 'g')
hold off

clear upperspec;clear lowerspec;

pause
clg

rec_upper=recoverm(upperfft,'ideallow',HZ,bitrate); %recover each signal
clear upperfft; %via a lowpass filter
rec_lower=recoverm(lowerfft,'ideallow',HZ,bitrate);
clear lowerfft;

axis(big_axis);

%Plot 32

subplot(211), %plot recovered signals against the input odd and even signals
plot(t,[rec_upper,nrzlodd])
title('Plot 32 - recovered and original odd bit signals')
xlabel('Time')
ylabel('Amplitude')

%Plot 33

```

```

subplot(212),
plot(t,[rec_lower,nrzleven])
title('Plot 33 - recovered and original even bit signals')
xlabel('Time')
ylabel('Amplitude')

pause

clg

comb_sig=par_ser(rec_upper,rec_lower,delta_t,bitrate); %put signal through
clear rec_upper;clear rec_lower; %parallel-to-serial

%Plot 34
                                %converter

subplot(211),
plot(t,[comb_sig,nrzlsig])
title('Plot 34 - combined odd and even recovered signals')
xlabel('Time')
ylabel('Amplitude')

axis;

subplot(212),
title('NO PLOT HERE')

```

**This page is intentionally
left blank.**

APPENDIX C—PROGRAMMING LABORATORIES

Name: _____

Section: _____

EO 3513 Programming Laboratory 1 *Signal and Spectrum Generation*

This laboratory introduces the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

M-file and **plot** instructions refer to building MATLAB script files; **questions** can be answered separately. Develop script files, produce and label plots, and answer questions as directed by your instructor. All plots should be numbered and titled, with x- and y-axes labeled.

Part 1—Produce and plot signals

A. Establish a time vector

Since your signals will be functions of time, a time vector must be established prior to generating a signal. Below is a sample time vector "t" that is one second in duration and has a "step size" of one ten-thousandth of a second. The variable name "delta_t" (Δt) is usually assigned to the step size. The time vector consists of 10,001 values, or points, starting at 0 and ending at 1.

`t=0:0.0001:1; %time vector`

M-file: Clear variables and graphs with "clear" and "clg."

Generate two time vectors, t1 and t2:

t1 with a duration of 1 second and step size of .001

t2 with a duration of 1 second and step size of .0001

B. Generate a signal

A single-tone periodic signal can be generated using the following formula, in which "f" represents frequency in Hz and "t" represents a time vector:

`s=cos(2*pi*f*t); %single-tone signal`

A multi-tone signal can be generated by adding sinusoids together. The following signal contains frequencies of 200, 500, and 800 Hz:

```
s3=10*cos(2*pi*200*t)+4*cos(2*pi*500*t)+6*cos(2*pi*800*t); %multi-tone signal
```

In future laboratories you will determine the maximum amplitude of a signal in order to calculate its power in the time domain. The signal "s3" has a maximum amplitude of 20 (in this case, conveniently found by adding the maximum amplitudes of its three cosines). If it is not easily determined from the formula or the signal plot, use the "max" command in MATLAB to find the approximate maximum signal amplitude:

```
max_of_s=max(s); %maximum amplitude in the signal s
```

Since the signal is a function of the time vector "t", it will be the same length as "t"—10,001 points.

M-file: **Generate three signals, s1, s2, and s3, using your two time vectors:**
 s1, function of t1, single-tone with frequency of less than 50
 s2, function of t2, single-tone with frequency of less than 500
 s3, function of t2, multi-tone with frequencies of less than 500

C. Controlling signal plots

The "plot", "title", "xlabel", and "ylabel" commands will produce clearly labeled graphs of your signals:

```
plot(t1,s1)
title('Signal')
xlabel('Time')
ylabel('Amplitude')
```

Other useful commands:

"Subplot" permits up to four plots per graphics window. The first digit of the argument represents the number of rows (up to two), the second digit the number of columns (up to two), and the third digit the placement of the graph (up to four). In the example below, a graph is placed in the lower right corner of the graphics window:

```
subplot(224), %formatted for 2 rows and 2 columns, in position 4
plot(t,s)
```

If you use the subplot command to position the first graph in the window, MATLAB will continue to follow that format until the graph window is filled.

"Pause" with no argument delays program execution until the user presses return (preventing graphs from whizzing by unobserved).

"Clg" between sets of plots prevents plots from being superimposed.

Plot 1: Plot s1.

Signal characteristics may be hard to distinguish when the entire signal vector is plotted. Described below are two methods of limiting plot size to 1000 points.

The first method involves restricting the number of points plotted. This method has an advantage in that the two vectors need not be the same length.

```
plot(t2(1:1000),s2(1:1000))
title('Signal')
xlabel('Time')
ylabel('Amplitude')
```

A second method is more tricky. Establish the axis parameters in vector format, then use the "axis" command to freeze the axes. Following use of the axis freeze, release the axis by typing the command "axis". (Note: Using the "hold off" command also unfreezes the axis.)

```
small_axis=[0 0.1 -20 20]; %min x, max x, min y, max y for axis freeze
axis(small_axis);
plot(t2,s2)
title('Signal')
xlabel('Time')
ylabel('Amplitude')
```

%continue to plot signals of similar scale, then release axis

axis; %don't forget to release axis!

Plot 2: Plot the first 1000 points of s2, using one of the methods described above.

You will often be asked to plot one signal over another in order to compare signals (a recovered signal and its message signal, for example). Three methods are described below.

The easiest method involves listing pairs of x and y arguments for the "plot" command:

```
plot(t2(1:500),s2(1:500),t2(1:500),s3(1:500)) %plot s3 over s2
title('Signal')
xlabel('Time')
ylabel('Amplitude')
```

A second method uses the "hold on" and "hold off" commands. "Hold on" freezes the current graph while plot commands are repeated. The example below also shows the use of the "pause" command, with the second signal drawn after a 3-second delay. When

using "pause" in conjunction with "hold," designate a color other than red in order to see both signals easily.

```
plot(t2(3501:4500),s3(3501:4500)) %plot signal with larger amplitude first
title('Signal')
xlabel('Time')
ylabel('Amplitude')
hold on
pause(3)
plot(t2(3501:4500),s2(3501:4500),'g')
hold off
```

In the third method, a signal matrix is created and plotted against a single time vector:

```
plot(t2(1:1000),[s2(1:1000);s3(1:1000)])
title('Signal')
xlabel('Time')
ylabel('Amplitude')
```

Plot 3: Plot s2 and s3 against t2, using one of the methods described above.

D. Printing plots

On PC platforms, the command "meta" creates a graphics file. "Prts" dumps the current graph window to a printer. "Print" sends a high-resolution copy to the printer. (The capability to use these commands may vary according to machine and software configuration.)

On Macintosh platforms, choose "print" from the "file" menu to print the active graph window, or use the "save as" command in the "file" menu to create a Quick-Draw graphics file.

Part 2—Produce and plot spectra

A. Calling a function

The Communications Toolbox contains the function "spectral," which has the following function call:

```
[specsig,HZ,fftsig]=spectral(s,delta_t);
```

"Spectral" produces a one-sided spectrum. To create a two-sided spectrum with relatively-correct amplitudes, use the following command:

```
two_sided_spec=abs(fftfshift(fftsig)); %plot the vector against its index
```

(Note: the two-sided spectrum above can not be plotted against the Hz vector returned from "spectral.")

The step size and number of points in the time vector both affect the vector length, which relates directly to the resolution of the spectrum. The exact relationship can be described as

$$\Delta f = \frac{1}{N * \Delta t}$$

where Δf represent the frequency resolution in Hz; N represents the number of points in the time vector, and Δt represents the difference between points in the time vector.

The step size alone affects the number of frequencies shown in the spectrum. The above vector of 10,001 points with $\Delta t=0.0001$ produces a one-sided spectrum of 5000 Hz with a resolution of 1 Hz. A vector with a step size of 0.001 will produce a one-sided spectrum of 500 Hz, regardless of its duration in seconds.

The vector length and step size above are appropriate for most of the signals generated in this laboratory set. To increase processing speed while drafting a script, reduce the vector length by reducing the duration of the time vector (from 1 second to 0.2 seconds, for example).

To call the function "spectral," pass in the input parameters "s" and "delta_t". The function outputs are the vector "specs" representing the spectrum of "s," the vector "Hz" for use as the x-axis when plotting the spectrum, and the vector "ffsig" for use in recovering the signal.

You will not always need to furnish all of the inputs to a function, nor require all of the outputs generated by a function. For example, you will not be recovering signals in this laboratory, and will not need "ffsig." Your function call to "spectral" might look like this:

```
[spectrum_1,Hz]=spectral(s1,0.001); %function call to spectral for s1
```

Vector lengths for spectral plots are usually not reduced in order to see as many of the spectral components as possible. A spectral plot might be generated as follows:

```
plot(Hz,spectrum_1)
title('Spectrum of s1')
xlabel('Frequency in Hz')
ylabel('Amplitude')
```

M-file: Using spectral.m, generate the spectrum for s1.

Plot 4: Plot the spectrum for s1.

Label the Hz value and the amplitude of the spectral component.

M-file: Using spectral.m, generate the spectrum for s2.

Plot 5: Plot the spectrum for s2.

Label the Hz value and the amplitude of the spectral component.

M-file: Using spectral.m, generate the spectrum for s3.

Plot 6: Plot the spectrum for s3.

Label the Hz values and the amplitudes of the spectral components.

Question 1: Compare Plots 4 and 5. Why does Plot 5 display more frequencies than Plot 4?

Question 2: Given the spectral plot of a single-tone signal, how could you determine the maximum amplitude and frequency for the signal?

Name: _____

Section: _____

EO 3513 Programming Laboratory 2A

Natural Sampling and Recovery

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

M-file and **plot** instructions refer to building MATLAB script files; **questions** can be answered separately. Develop script files, produce and label plots, and answer questions as directed by your instructor. All plots should be numbered and titled, with x- and y-axes labeled.

Part 1—Generate a naturally-sampled signal and its spectrum

A. Generate a signal

M-file: Establish a time vector with a Δt of 0.0001 and a duration of 1 second.

Generate a multi-tone message signal with frequencies less than 500. (Tones should have the same amplitude.)

Question 1: What is the maximum amplitude of the signal?

Question 2: What is the highest frequency in the signal? What is the Nyquist rate?

B. Naturally-sample the signal

M-file: Use `natsamp.m` to sample the signal. Use a sampling rate above the Nyquist rate but not more than 2000 Hz.

Plot 1: Plot the sampled signal over the message signal.

Question 3: Calculate the following values for the sampled signal:

sampling period T (calculated in seconds)
pulse duration τ (calculated in seconds)

Describe the pulse shape of the sampled signal.

C. Generate the spectrum

M-file: Use spectral.m to generate the spectrum.

Plot 2: Plot the spectrum of the sampled signal.

Label the following groups of frequencies in the spectrum:

baseband signal frequencies
spectral components associated with the sampling
frequency (f_s) and each of its multiples ($2f_s$, $3f_s$, etc.)

The amplitude spectrum of a naturally-sampled signal can be determined using the formula

$$X_s(f) = \sum_{N=0}^{\infty} P_N X(f - Nf_s)$$

where

$$P_N = \frac{d \sin(N\pi d)}{N\pi d}$$

and d is the duty cycle and N indicates the number of the harmonic.

Question 4: Calculate P_N for $N = 1$, $N = 2$, and $N = 3$.

Compare with the values shown on the spectral plot.

Question 5: Describe the overall shape of the spectrum. Does the spectrum conform to your theoretical expectations? Note any discrepancies.

Part 2—Recover the message signal

A. Recover the message signal

The function “recovers” calls the filtering function designated in the input parameter. The Communications Toolbox contains two lowpass filters, “idealowl” and “lowpass.” Select a cutoff frequency that captures only the baseband signal frequencies.

M-file: Use `recovers.m` to recover the message signal.

Plot 3: Plot the recovered signal over the message signal.

M-file: To free memory, clear all variables other than the time vector and message signal.

Part 3—Observe the effects of aliasing

A. Produce an undersampled signal

M-file: Use `natsamp.m` to sample the message signal at less than the Nyquist rate (twice the highest frequency in the signal).

Plot 4: Plot the undersampled signal over the message signal.

M-file: Use `spectral.m` to generate the spectrum of the undersampled signal.

Plot 5: Plot the spectrum of the undersampled signal.

Question 6: Compare Plot 5 with Plot 2. What is the effect of undersampling on the spectrum?

M-file: Use `recovers.m` to recover the undersampled signal. Use the same cutoff frequency for the lowpass filter that you used in Part 2.

Plot 6: Plot the recovered undersampled signal over the message signal.

Label the recovered signal.

Question 7: What is the effect of undersampling on signal recovery?

M-file: To free memory, clear all variables other than the time vector and message signal.

Part 4—Observe the effect on the spectrum of varying the duty cycle

A. Generate the sampled signal

M-file: Use natsamp.m to sample the message signal at the same rate used in Part 1b, varying the duty cycle.

Plot 7: Plot the sampled signal.

B. Generate the spectrum

M-file: Use spectral.m to generate the spectrum.

Plot 8: Plot the spectrum of the sampled signal.

Label the Hz value at the first zero crossing ($1/\tau$).

Question 8: Compare Plot 8 with Plot 2. What is the effect of changing the duty cycle on the sampled signal baseband bandwidth?

Name: _____

Section: _____

EO 3513 Programming Laboratory 2B

Flattop Sampling and Recovery

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

M-file and plot instructions refer to building MATLAB script files; questions can be answered separately. Develop script files, produce and label plots, and answer questions as directed by your instructor. All plots should be numbered and titled, with x- and y-axes labeled.

Part 1—Generate a flattop-sampled signal and its spectrum

A. Generate a signal

M-file: Establish a time vector with a Δt of 0.0001 and a duration of 1 second.

Generate a multi-tone message signal with frequencies less than 500. (Tones should have the same amplitude.)

Question 1: What is the maximum amplitude of the signal?

Question 2: What is the highest frequency in the signal? What is the Nyquist rate?

B. Flattop-sample the signal

M-file: Use flattop.m to sample the signal. Use a sampling rate above the Nyquist rate but not more 2000 Hz.

Plot 1: Plot the sampled signal over the message signal.

Question 3: Calculate the following values for the sampled signal:

sampling period T (calculated in seconds)
pulse duration τ (calculated in seconds)

Describe the pulse shape of the sampled signal.

C. Generate the spectrum

M-file: Use spectral.m to generate the spectrum.

Plot 2: Plot the spectrum of the sampled signal.

Label the following groups of frequencies in the spectrum:

baseband signal frequencies
spectral components associated with the sampling
frequency (f_s) and each of its multiples ($2f_s$, $3f_s$, etc.)

The amplitude spectrum of a flattop-sampled signal can be determined using the formula

$$X_s(f) = \sum_{N=0}^{\infty} P_N X(f - Nf_s)$$

where

$$P_N = \frac{d \sin(N\tau f)}{N\tau f}$$

and d is the duty cycle, τ is the pulse duration in seconds, and f indicates the frequency in Hz.

Question 4: Calculate P_N for $f = 550$, $f = 700$, and $f = 880$.

Compare with the values shown on the spectral plot.

Question 5: Describe the overall shape of the spectrum. Does the spectrum conform to your theoretical expectations? Note any discrepancies.

Part 2—Recover the message signal

A. Recover the message signal

The function “recovers” calls the filtering function designated in the input parameter. The Communications Toolbox contains two lowpass filters, “ideallow” and “lowpass.” Select a cutoff frequency that captures only the baseband signal frequencies.

M-file: Use `recovers.m` to recover the message signal.

Plot 3: Plot the recovered signal over the message signal.

M-file: To free memory, clear all variables other than the time vector and message signal.

Part 3—Observe the effects of aliasing

A. Produce an undersampled signal

M-file: Use `flattop.m` to sample the message signal at less than the Nyquist rate (twice the highest frequency in the signal).

Plot 4: Plot the undersampled signal over the message signal.

M-file: Use `spectral.m` to generate the spectrum of the undersampled signal.

Plot 5: Plot the spectrum of the undersampled signal.

Question 6: Compare Plot 5 with Plot 2. What is the effect of undersampling on the spectrum?

M-file: Use `recovers.m` to recover the undersampled signal. Use the same cutoff frequency for the lowpass filter that you used in Part 2.

Plot 6: Plot the recovered undersampled signal over the message signal.

Label the recovered signal.

Question 7: What is the effect of undersampling on signal recovery?

M-file: To free memory, clear all variables other than the time vector and message signal.

Part 4—Observe the effect on the spectrum of varying the duty cycle

A. Generate a sampled signal

M-file: Use `flattop.m` to sample the message signal at the same rate used in Part 1b, varying the duty cycle.

Plot 7: Plot the sampled signal.

B. Generate the spectrum

M-file: Use `spectral.m` to generate the spectrum.

Plot 8: Plot the spectrum of the sampled signal.

Label the Hz value at the first zero crossing ($1/\tau$).

Question 8: Compare Plot 8 with Plot 2. What is the effect of changing the duty cycle on the sampled signal baseband bandwidth?

Name: _____

Section: _____

EO 3513 Programming Laboratory 2C

Impulse Sampling and Recovery

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

M-file and plot instructions refer to building MATLAB script files; questions can be answered separately. Develop script files, produce and label plots, and answer questions as directed by your instructor. All plots should be numbered and titled, with x- and y-axes labeled.

Part 1—Generate a impulse-sampled signal and its spectrum

A. Generate a signal

M-file: Establish a time vector with a Δt of 0.0001 and a duration of 1 second.

Generate a multi-tone message signal with frequencies less than 500. (Tones should have the same amplitude.)

Question 1: What is the maximum amplitude of the signal?

Question 2: What is the highest frequency in the signal? What is the Nyquist rate?

B. Impulse-sample the signal

M-file: Use `imsamp.m` to sample the signal. Use a sampling rate above the Nyquist rate but not more than 2000 Hz.

Plot 1: Plot the sampled signal over the message signal.

Question 3: Calculate the duration of the sampling period T (in seconds).

C. Generate the spectrum

M-file: Use `spectral.m` to generate the spectrum.

Plot 2: Plot the spectrum of the sampled signal.

Label the following groups of frequencies in the spectrum:

baseband signal frequencies
spectral components associated with the sampling
frequency (f_s) and each of its multiples ($2f_s$, $3f_s$, etc.)

Question 4: Describe the overall shape of the spectrum. Does the spectrum conform to your theoretical expectations? Note any discrepancies.

Part 2—Recover the message signal

A. Recover the message signal

The function “recovers” calls the filtering function designated in the input parameter. The Communications Toolbox contains two lowpass filters, “`idealowl`” and “`lowpass`.” Select a cutoff frequency that captures only the baseband signal frequencies.

M-file: Use `recovers.m` to recover the impulse-sampled signal.

Plot 3: Plot the recovered signal over the message signal.

M-file: To free memory, clear all variables other than the time vector and message signal.

Part 3—Observe the effects of aliasing

A. Produce an undersampled signal

M-file: Use `impsamp.m` to sample the message signal at less than the Nyquist rate (twice the highest frequency in the signal).

Plot 4: Plot the undersampled signal over the message signal.

M-file: Use `spectral.m` to generate the spectrum of the undersampled signal.

Plot 5: Plot the spectrum of the undersampled signal.

Question 5: Compare Plot 5 with Plot 2. What is the effect of undersampling on the spectrum?

M-file: Use `recovers.m` to recover the undersampled signal. Use the same cutoff frequency for the lowpass filter that you used in Part 2.

Plot 6: Plot the recovered undersampled signal over the message signal.

Label the recovered signal.

Question 6: What is the effect of undersampling on signal recovery?

**This page is intentionally
left blank.**

Name: _____

Section: _____

EO 3513 Programming Laboratory 3A

Pulse Modulation (PAM and PWM)

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

M-file and **plot** instructions refer to building MATLAB script files; **questions** can be answered separately. Develop script files, produce and label plots, and answer questions as directed by your instructor. All plots should be numbered and titled, with x- and y-axes labeled.

Part 1—Observe the differences in the time domain for two types of pulse-modulated signals (PAM and PWM)

A. Generate a signal

M-file: Establish a time vector with a Δt of 0.0001 and a duration of 1 second.

Generate a multi-tone message signal with frequencies less than 200.

Question 1: What is the maximum amplitude of the message signal?

B. Modulating the signal

You studied the flat-top-sampled signal in Laboratory 2, and saw that while its pulses varied in amplitude, they always appeared at the start of the sampling period T , and had a constant duration τ . Flat-top sampling is one implementation of pulse-amplitude modulation (natural sampling is the other). In this laboratory, the familiar characteristics of the pulse-amplitude modulated (PAM) signal will be compared to characteristics of the pulse-width modulated signal.

M-file: Use `flat_top.m` to pulse-amplitude-modulate the signal. Use a sampling rate of 500 Hz.

Plot 1: Plot the PAM signal over the message signal. Show a portion of the signal that contains the maximum signal amplitude.

Question 2: Calculate the following values, in seconds, for the PAM signal:

sampling period T
pulse duration τ

Like PAM signals, pulse-width modulated (PWM) signal pulses begin with the sampling period T . Pulse amplitudes are constant, while their widths vary, based on the amplitude of the message signal at each pulse beginning. To facilitate this variation in pulse width, the maximum pulse duration is expressed as a fraction of the sampling period T . The widest pulse that could occur would be this fraction of the sampling period T , located where the maximum signal amplitude fell at the beginning of a pulse. The most narrow pulse would occur where the minimum signal amplitude fell at the beginning of a pulse.

M-file: Use `pulsewid.m` to pulse-width-modulate the signal. Use a sampling rate of 500 Hz and a maximum pulse duration of greater than 0.5.

Plot 2: Plot the PWM signal over the message signal. Show the same portion of the signal (containing the maximum signal amplitude) as displayed in Plot 1.

Question 3: What is the maximum pulse duration that could occur in your PWM signal?

Part 2—Observe the differences in the frequency domain for the two types of modulation

A. Generate the spectra

M-file: Use `spectral.m` to generate the spectrum of the PAM signal.

Plot 3: Plot the spectrum of the PAM signal.

Label the following values:

Hz values of the baseband signal frequencies
sampling frequency (fs)

M-file: Use `spectral.m` to generate the spectrum of the PWM signal.

Plot 4: Plot the spectrum of the PWM signal.

Label the Hz value of the sampling frequency (f_s).

B. Calculating baseband bandwidth

Recall that while PAM signals have a baseband bandwidth of approximately $0.5/\tau$, PWM signals have larger baseband bandwidths, approximately $0.5/\text{risetime}$. Consider the risetime of the pulses in your modulated signals to be equal to the step size in the time vector.

Question 4: Using the above approximations, calculate the baseband bandwidths for the PAM and PPM signals. Do these values reflect what you observe in the spectral plots? Note any discrepancies.

**This page is intentionally
left blank.**

Name: _____

Section: _____

EO 3513 Programming Laboratory 3B

Pulse Modulation (PAM and PPM)

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

M-file and **plot** instructions refer to building MATLAB script files; **questions** can be answered separately. Develop script files, produce and label plots, and answer questions as directed by your instructor. All plots should be numbered and titled, with x- and y-axes labeled.

Part 1—Observe the differences in the time domain for two types of pulse-modulated signals (PAM and PPM)

A. Generate a signal

M-file: Establish a time vector with a Δt of 0.0001 and a duration of 1 second.

Generate a multi-tone message signal with frequencies less than 200.

Question 1: What is the maximum amplitude of the message signal?

B. Modulating the signal

You studied the flat-top-sampled signal in Laboratory 2, and saw that while its pulses varied in amplitude, they always appeared at the start of the sampling period T , and had a constant duration τ . Flat-top sampling is one implementation of pulse-amplitude modulation (natural sampling is the other). In this laboratory, the familiar characteristics of the pulse-amplitude modulated (PAM) signal will be compared to characteristics of the pulse-position modulated signal.

M-file: Use `flat_top.m` to pulse-amplitude-modulate the signal. Use a sampling rate of 500 Hz.

Plot 1: Plot the PAM signal over the message signal. Show a portion of the signal that contains the maximum signal amplitude.

Question 2: Calculate the following values, in seconds, for the PAM signal:

sampling period T
pulse duration τ

A pulse-position modulated (PPM) signal, like a pulse-amplitude modulated signal, has a constant duration τ . However, the pulse beginnings vary in location within the sampling period T . Most PPM systems vary pulse position from the middle of the sampling period T . Negative signal amplitudes cause the pulse to shift left; positive signal amplitudes cause the pulse to shift right.

The function "pulspos.m" transmits signal information via the amount of the pulse offset from the beginning of the sampling period. To allow for larger variations in the pulse offset, the pulse duration τ is kept small. For a PPM signal with a duty cycle of 0.2, the largest pulse offset that could occur would be 0.8 of the sampling period T , observed at the maximum signal amplitude. No pulse offset occurs at the minimum signal amplitude.

M-file: Use `pulspos.m` to pulse-position-modulate the signal. Use a sampling rate of 500 Hz and a pulse duration of less than 0.5.

Plot 2: Plot the PPM signal over the message signal. Show the same portion of the signal (containing the maximum signal amplitude) as displayed in Plot 1. (Note: Using the "grid" command may help identify the boundaries of the sampling periods.)

Question 3: What is the largest pulse offset that could occur in your PPM signal?

Part 2—Observe the differences in the frequency domain for the two types of modulation

A. Generate the spectra

M-file: Use `spectral.m` to generate the spectrum of the PAM signal.

Plot 3: Plot the spectrum of the PAM signal.

Label the following values:

Hz values of the baseband signal frequencies
sampling frequency (fs)

M-file: Use spectral.m to generate the spectrum of the PPM signal.

Plot 4: Plot the spectrum of the PPM signal.

Label the Hz value of the sampling frequency (fs).

B. Calculating baseband bandwidth

Recall that while PAM signals have a baseband bandwidth of approximately $0.5/\tau$, PPM signals have larger baseband bandwidths, approximately $0.5/\text{risetime}$. Consider the risetime of the pulses in your modulated signals to be equal to the step size in the time vector.

Question 4: Using the above approximations, calculate the baseband bandwidths for the PAM and PPM signals. Do these values reflect what you observe in the spectral plots? Note any discrepancies.

**This page is intentionally
left blank.**

Name: _____

Section: _____

EO 3513 Programming Laboratory 4A *Analog-to-Digital Conversion (Quantization)*

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

M-file and **plot** instructions refer to building MATLAB script files; **questions** can be answered separately. Develop script files, produce and label plots, and answer questions as directed by your instructor. All plots should be numbered and titled, with x- and y-axes labeled.

Part 1—Evaluate two analog-to-digital converters

A. Evaluate a bipolar converter

The function "quantize.m" is used to set the quantization characteristic for a bipolar quantization system. "Quantize.m" converts signals with values between -10 and +10 volts.

M-file: Use quantize.m to generate the characteristic for a 3-bit bipolar analog-to-digital converter.

Plot 1: Plot the quantization characteristic, using the output vector quanch_x to represent voltage in, and quanch_y to represent voltage out.

Question 1: Calculate the following values relating to the quantization characteristic for the 3-bit bipolar converter:

dynamic range
actual step size
actual resolution
percentage resolution
number of levels

B. Evaluate a unipolar converter

"Quantuni.m" is a unipolar quantizing function that converts signals between values of 0 and +10 volts.

M-file: Use quantuni.m to generate the characteristic for a 5-bit unipolar analog-to-digital converter.

Plot 2: Plot the quantization characteristic, using the output vector quanch_x to represent voltage in, and quanch_y to represent voltage out.

Question 2: Calculate the following values relating to the quantization characteristic for the 5-bit unipolar converter:

dynamic range
actual step size
actual resolution
percentage resolution
number of levels

Part 2—Observe the quantization process

A. Generate and sample a signal

M-file: Establish a time vector with a Δt of 0.0001 and a duration of 0.1 seconds.

Generate a multi-tone message signal s with frequencies less than 50. (Maximum signal amplitude should not exceed 10.)

Signals are typically sampled prior to quantizing.

M-file: Use flattop.m to sample the signal, using a sampling rate of 500 Hz or greater.

B. Quantize the signal using a 3-bit bipolar converter

M-file: Use quantize.m to quantize the sampled signal as if using a 3-bit bipolar converter. Use the same sampling rate for sampling and quantizing. (The characteristic of this converter is displayed in Plot 1.)

Plot 3: Plot the 3-bit bipolar quantized signal over the sampled signal.

C. Quantize the signal using a 5-bit unipolar converter

M-file: Use `quantize.m` to quantize the sampled signal as if using a 5-bit bipolar converter.

Plot 4: Plot the 5-bit bipolar quantized signal over the sampled signal.

A bin number representing voltage level is assigned to each sample in the quantized signal. Bin numbers begin with 0.

D. Relate bin numbers to voltage levels

M-file: Print the values of `quanch_y` for the 3-bit bipolar converter (the highest "voltage" level appears twice, for plotting purposes only.). Then print the first 5 values of `bin_nums`.

Question 3: List the amplitude ("voltage") of the 3-bit bipolar quantized signal in each of the first 5 sampling periods.

M-file: Print the values of `quanch_y` for the 5-bit bipolar converter; then print the first 5 values of `bin_nums`.

Question 4: List the amplitude ("voltage") of the 5-bit bipolar quantized signal in each of the first 5 sampling periods.

Part 3—Measure the quantization noise

The difference between the message signal and the quantized signal is referred to as "quantization noise"—noise introduced by the quantization process. The function "`snr.m`" returns the signal to noise ratio, measured in dB.

A. Find the signal to noise ratios

Plot 5: Plot the 3-bit bipolar quantized signal over the message signal.

M-file: Use `snr.m` to find the signal to noise ratio for the message signal and the 3-bit bipolar quantized signal.

Plot 6: Plot the 5-bit bipolar quantized signal over the message signal.

M-file: Use `snr.m` to find the signal to noise ratio for the message signal and the 5-bit bipolar quantized signal.

B. Compare quantization noise for the two systems

Question 5: List the values of the signal to noise ratios for the 3-bit bipolar and 5-bit bipolar quantized signals. Which converter produced less quantization noise?

Name: _____

Section: _____

EO 3513 Programming Laboratory 4B

Pulse Code Modulation (PCM)

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

M-file and plot instructions refer to building MATLAB script files; **questions** can be answered separately. Develop script files, produce and label plots, and answer questions as directed by your instructor. All plots should be numbered and titled, with x- and y-axes labeled.

Part 1—Generate a bitstream to encode

A. Generate a message signal

M-file: Establish a time vector with a Δt of 0.0001 and a duration of 1 second.

Generate a multi-tone message signal s with frequencies less than 50. (Maximum signal amplitude should not exceed 10).

B. Quantize the message signal

M-file: Use `quantize.m` to quantize the message signal, using 2 symbols and from 3 to 6 elements. Use a sampling rate of 100 to 200 Hz.

Plot 1: Plot the quantized signal over the message signal.

C. Generate a bitstream

The function "encode.m" will convert "bin_nums," the vector returned from "quantize.m" containing the quantization levels, to "codedsig," a bitstream of 1's and 0's. Each value in bin_nums is represented by one binary word in "codedsig." The number of bits in each word is determined by the number of elements.

M-file: Use `encode.m` to convert "bin_nums" to a bitstream.

Part 2—Encode the bitstream using pulse code modulation (PCM)

The functions “nrzluni.m,” “rzuni.m,” and “manchest.m” translate the vector “codedsig” into PCM signals. The bit rate for these PCM-encoded signals is calculated by multiplying the sampling rate by the number of elements.

A. Generate a non-return-to-zero level (NRZL) unipolar coded signal and spectrum

M-file: Using nrzlevel.m, generate a NRZL unipolar coded signal.

Plot 2: Plot an expanded view of the NRZL unipolar coded signal over the quantized signal, showing the first 8 to 10 words.

Question 1: In your own words, describe the NRZL unipolar encoding scheme.

M-file: Using spectral.m, generate a NRZL unipolar coded signal spectrum.

Plot 3: Plot the NRZL unipolar coded signal spectrum.

Question 2: Predict an adequate baseband signal bandwidth for the NRZL unipolar coded signal based on its spectral plot.

M-file: To free memory, clear the variables for the NRZL unipolar coded signal and spectrum.

B. Generate a return-to-zero level (RZL) unipolar coded signal and spectrum

M-file: Using rzuni.m, generate a RZL unipolar coded signal.

Plot 4: Plot the RZL unipolar coded signal over the quantized signal, showing the same portion of the signal as in Plot 5.

Question 3: In your own words, describe the RZL unipolar encoding scheme.

M-file: Using spectral.m, generate an RZL unipolar coded signal spectrum.

Plot 5: Plot the RZL unipolar coded signal spectrum.

Question 4: Predict an adequate baseband signal bandwidth for the RZL unipolar coded signal based on its spectral plot.

M-file: To free memory, clear the variables for the RZL unipolar coded signal and spectrum.

C. Generate a manchester coded signal and spectrum

M-file: Using manchest.m, generate a manchester coded signal.

Plot 6: Plot the manchester coded signal over the quantized signal, showing the same portion of the signal as in Plot 5.

Question 5: In your own words, describe the manchester encoding scheme.

M-file: Using spectral.m, generate a manchester coded signal spectrum.

Plot 7: Plot the manchester coded signal spectrum.

Question 6: Predict an adequate baseband signal bandwidth for the manchester coded signal based on its spectral plot.

M-file: Print out the values of "codedsig" that represent the bits shown in Plots 2, 4, and 6.

Question 7: Record the values of "codedsig." Is this bit pattern reflected on Plots 2, 4, and 6?

D. Estimate bandwidth for the PCM signals

The minimum theoretical PCM bandwidth for sinc-shaped pulses is $B*N$, the baseband message signal bandwidth times the number of elements(bits). Rectangular pulses theoretically require an *infinite* bandwidth, but can be estimated based on τ , the pulse duration:

$$B = 0.5/\tau$$

The value of τ depends on the PCM encoding scheme employed. For NRZL coded signals, τ is equal to the bit duration. For RZL and manchester coded signals, τ is equal to 1/2 of the bit duration.

Question 8: Calculate the approximate baseband bandwidth for the PCM signals:

NRZL unipolar coded signal
RZL and manchester coded signals

How do these values compare with your predictions?

**This page is intentionally
left blank.**

Name: _____

Section: _____

EO 3513 Programming Laboratory 4C *Companding*

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

M-file and plot instructions refer to building MATLAB script files; questions can be answered separately. Develop script files, produce and label plots, and answer questions as directed by your instructor. All plots should be numbered and titled, with x- and y-axes labeled

Part 1—Observe the effects of companding on the message signal

A. Generate the signal

M-file: Establish a time vector with a Δt of 0.0001 and a duration of 0.05 seconds.

Generate the following multi-tone message signal s :

```
s=2+2.1*cos(2*pi*50*t)+1.7*cos(4*pi*50*t)...  
+1.5*cos(6*pi*50*t)+1.3*cos(8*pi*50*t);
```

B. Compare compression characteristics for high and low values of μ

Companding (the process of compressing, then expanding) improves the quantization process by proportioning signals that spend most of the time in the lower range of the dynamic range. The functions "compress.m" and "expand.m" simulate a μ -255 compander. Values of μ range from 1 to 255. An input of $\mu = 255$ produces the maximum compression and expansion.

When using "compress.m" and "expand.m," be sure pass in the exact maximum of each individual signal. Use the "max" command as part of the parameter.

M-file: Using compress.m, compress the message signal, using $\mu = 255$.

Using `expand.m`, expand the compressed signal, using $\mu = 255$.

Plot 1: Plot the compressed signal ($\mu = 255$) over the message signal. (Notice the low-level signal activity.)

Plot the expanded signal ($\mu = 255$) on the same graph. The expanded signal should plot directly over the message signal.

M-file: Using `compress.m`, compress the message signal, using $\mu = 5$.

Using `expand.m`, expand the compressed signal, using $\mu = 5$.

Plot 2: Plot the compressed signal ($\mu = 5$) over the message signal.

Plot the expanded signal ($\mu = 5$) on the same graph. The expanded signal should plot directly over the message signal.

Question 1: What is the effect on the compressed signal of increasing the value of μ ?

Part 2—Reduce quantization noise by companding the message signal

Your goal in this section of the laboratory is to design a companding system that will reduce quantization noise. The signal s is given, as is the use of a 2-bit unipolar quantizer. Your variables are the sampling rate and the value of μ . (Note: Your variables must remain constant within the system, but may be adjusted to represent different systems.)

“`Quantuni.m`” employs a truncation rather than a rounding scheme, which tends to increase quantization noise.

A. Sample and quantize the message signal

M-file: Use `flattop.m` to sample the message signal.

M-file: Use `quantuni.m` to quantize the sampled signal (use the same sampling rate for sampling and quantizing). Pass in 2 symbols and 2 elements.

Plot 3: Plot the quantized signal over the sampled signal.

Plot 4: Plot the message signal over the quantized signal. (This graph illustrates the degree of quantization noise present without companding.)

B. Establish a benchmark value for signal to noise ratio

M-file: Use `snr.m` to calculate the signal to noise ratio for the message and quantized signals (no companding).

Question 2: Obtain the value of the signal to noise ratio (no companding). Record this value. By exceeding this benchmark value, you will be decreasing quantization noise.

C. Compress, sample, and quantize the message signal

M-file: Use `compress.m` to compress the message signal.

Plot 5: Plot the compressed signal over the message signal to gauge the degree of compression.

M-file: Use `flattop.m` to sample the compressed signal, using the same sampling rate as before.

M-file: Use `quantuni.m` to quantize the sampled compressed signal, passing in 2 symbols and 2 elements. Continue to use the same sampling rate.

Plot 6: Plot the quantized compressed signal over the sampled compressed signal.

D. Expand the quantized compressed signal

M-file: Use `expand.m` to expand the quantized compressed signal.

Plot 7: Plot the companded signal over the message signal. (This graph illustrates the degree of quantization noise with companding.)

E. Find the signal to noise ratio for the companded signal

M-file: Use `snr.m` to calculate the signal to noise ratio for the message and companded signals.

Question 3: Obtain the value of the signal to noise ratio using companding. Record this value.

If the signal to noise ratio has not increased over the benchmark value obtained in Question 2, examine the signals shown in Plot 7. Try to determine the major cause of the noise in the companded signal. Adjust one or both variables in your system, and repeat the companding steps.

Question 4: Provide the following characteristics of your system:

sampling rate
value of μ

Name: _____

Section: _____

EO 3513 Programming Laboratory 5

Amplitude Modulation Double Sideband (AM DSB)

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

M-file and **plot** instructions refer to building MATLAB script files; **questions** can be answered separately. Develop script files, produce and label plots, and answer questions as directed by your instructor. All plots should be numbered and titled, with x- and y-axes labeled.

Part 1—Generate single- and multi-tone message signals and spectra

A. Generate the message signals

M-file: Establish a time vector with a Δt of 0.0001 and a duration of 1 second.

Generate a single-tone signal with a frequency of less than 500.

Generate a multi-tone signal with frequencies of less than 500.

Plot 1: Plot the single-tone message signal.

Plot 2: Plot the multi-tone message signal.

B. Predict power and bandwidth for the message signals

Parseval's theorem states that average signal power can be calculated in either the time or the frequency domain. The following formula applies to calculation in the time domain:

$$P = A_0^2 + \frac{1}{2} \sum_{N=1}^{\infty} (A_N^2 + B_N^2)$$

Peak power is calculated as follows:

$$P_P = \frac{A_P^2}{2}$$

Question 1: Calculate the following values for the single-tone message signal:

peak power
average power
baseband bandwidth

C. Verify bandwidth for the message signals

M-file: Use spectral.m to generate the spectrum of the single-tone message signal.

Plot 3: Plot the spectrum of the single-tone signal.

Label the following values:

Hz value of the spectral component
amplitude of the spectral component
baseband signal bandwidth in Hz

M-file: Use spectral.m to generate the spectrum of the multi-tone message signal.

Plot 4: Plot the spectrum of the multi-tone signal.

Label the following values:

Hz value of the spectral component
amplitude of the spectral component
baseband signal bandwidth in Hz

M-file: To free memory, clear the variables representing the spectra of the single- and multi-tone signals.

Part 2—Generate amplitude modulated double sideband (AM DSB) signals using single- and multi-tone input

A. Generate the single- and multi-tone AM DSB signals

M-file: Modulate the single-tone signal by multiplying it with a cosine with a carrier frequency of 1500 to 3000 Hz. (Remember to use the “.” operator.)

Modulate the multi-tone signal by multiplying it with a cosine with a carrier frequency of 1500 to 3000 Hz (choose a different carrier frequency than used for single-tone modulation).

Plot 5: Plot the single-tone AM DSB signal.

Plot 6: Plot an expanded view of the single-tone AM DSB and message signals that shows one or more zero crossings.

Label the phase shifts shown in this portion of the signal.

Plot 7: Plot the multi-tone AM DSB signal.

Plot 8: Plot an expanded view of the multi-tone AM DSB and message signals that shows one or more zero crossings.

Label the phase shifts shown in this portion of the signal.

B. Predict power for the single-tone AM DSB signal

Peak power for the AM DSB signal is calculated as before:

$$P_p = \frac{A_p^2}{2}$$

Average power for the AM DSB signal is obtained using by adding the power produced by each of the two components, resulting in

$$P = \frac{A^2}{4}$$

Question 2: Predict the following values for the single-tone AM DSB signal:

peak power
average power
transmission bandwidth

C. Verify the bandwidth of the single- and multi-tone AM DSB signals

M-file: Use `spectral.m` to generate the spectrum of the single-tone AM DSB signal.

Use `spectral.m` to generate the spectrum of the multi-tone AM DSB signal.

Plot 9: Plot the spectrum of the single-tone AM DSB signal.

Label the following values:

amplitude of each spectral component
Hz value of each spectral component
transmission bandwidth in Hz

Plot 10: Plot the spectrum of the multi-tone AM DSB signal.

Label the following values:

amplitude of each spectral component
Hz value of each spectral component
transmission bandwidth in Hz

M-file: To free memory, clear variables representing spectra for the single- and multi-tone AM DSB signals.

D. Verify power for the single-tone AM DSB signal

Peak power is verified in the time domain.

M-file: To calculate peak power, use the “max” command to find the single-tone AM DSB signal maximum; then square this value and divide by 2.

Average power is verified in the frequency domain.

M-file: To calculate total average power, use `psd.m` to obtain a vector representing power spectral density of the single-tone AM DSB signal; then use the “sum” command to add the frequencies.

Question 3: Record the values representing peak and average power for the single-tone signal.

Do your calculations for bandwidth and power in Question 2 agree with the computer-generated values?

Part 3—Recover the AM DSB signals

A. Demodulate the AM DSB signals (multiply by their carriers)

M-file: **Multiply the single-tone AM DSB signal by its carrier.**

Use `spectral.m` to generate the spectrum of the demodulated single-tone signal.

Multiply the multi-tone AM DSB signal by its carrier.

Use `spectral.m` to generate the spectrum of the demodulated multi-tone signal.

To free memory, clear variables representing the single- and multi-tone AM DSB signals, and the demodulated single- and multi-tone AM DSB signals (all are time domain vectors).

Plot 11: **Plot the spectrum of the demodulated single-tone signal.**

Plot 12: **Plot the spectrum of the demodulated multi-tone signal.**

M-file: **To free memory, clear variables representing the spectra of the demodulated single- and multi-tone demodulated signals.**

B. Filter and recover the message signals

“Lowpass” and “ideallow” are the lowpass filters available. Choose cutoff frequencies that capture only the baseband signal frequencies.

M-file: **Use `recoverm.m` to recover and filter the single-tone baseband signal frequencies.**

Multiply the recovered signal by a factor of 2.

Use `recoverm.m` to recover and filter the multi-tone baseband signal frequencies.

Multiply the recovered signal by a factor of 2.

Plot 13: **Plot the recovered single-tone signal over the message signal.**

Plot 14: **Plot the recovered multi-tone signal over the message signal.**

Question 4: **Why is coherent detection (detection using the carrier) necessary for an AM DSB signal?**

**This page is intentionally
left blank.**

Name: _____

Section: _____

EO 3513 Programming Laboratory 6 *Amplitude Modulation Single Sideband (AM SSB)*

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

M-file and plot instructions refer to building MATLAB script files; questions can be answered separately. Develop script files, produce and label plots, and answer questions as directed by your instructor. All plots should be numbered and titled, with x- and y-axes labeled.

Part 1—Generate single- and multi-tone message signals and spectra

A. Generate the message signals

M-file: Establish a time vector with a Δt of 0.0001 and a duration of 1 second.

 Generate a single-tone signal with a frequency of less than 500.

 Generate a multi-tone signal with frequencies of less than 500.

Plot 1: Plot the single-tone message signal.

Plot 2: Plot the multi-tone message signal.

B. Predict the power and bandwidth for the message signals

Parseval's theorem states that average signal power can be calculated in either the time or the frequency domain. The following formula applies to calculation in the time domain:

$$P = A_0^2 + 1/2 \sum (A_n^2 + B_n^2), \text{ summation from } 1 \text{ to } \infty$$

Peak power is calculated as follows:

$$P_p = A_p^2 / 2$$

Question 1: Calculate the following values for the single-tone message signal:

peak power
average power
baseband bandwidth

C. Verify bandwidth for the message signals

M-file: Use spectral.m to generate the spectrum of the single-tone message signal.

Plot 3: Plot the spectrum of the single-tone signal.

Label the following values:

Hz value of the spectral component
amplitude of the spectral component
baseband signal bandwidth in Hz

M-file: Use spectral.m to generate the spectrum of the multi-tone message signal.

Plot 4: Plot the spectrum of the multi-tone signal.

Label the following values:

Hz value of the spectral component
amplitude of the spectral component
baseband signal bandwidth in Hz

M-file: To free memory, clear the variables representing the spectra of the single- and multi-tone message signals.

Part 2—Generate amplitude modulated single sideband (AM SSB) signals using single- and multi-tone input

A. Generate the single- and multi-tone AM SSB signals

Single sideband modulation could be accomplished, in theory, by double sideband modulation followed by filtering of unwanted frequencies. In practice, however, retaining

one sideband while rejecting the other is a complex procedure. In this laboratory, you will use a Hilbert transform, simulated by the function "hilbert.m," to apply a 90° phase shift to the signal, cancelling either the upper or lower sideband.

"Hilbert.m" is called by the function "ssb.m."

M-file: Use ssb.m to modulate the single-tone signal. Pass in a carrier frequency of 1500 to 3000 Hz.

Use ssb.m to modulate the multi-tone signal. Pass in a carrier frequency of 1500 to 3000 Hz (choose a different carrier frequency than used for single-tone modulation).

Plot 5: Plot the single-tone AM lower sideband (LSB) signal.

Plot 6: Plot an expanded view of the single-tone AM LSB and message signals.

Plot 7: Plot the single-tone AM upper sideband (USB) signal.

Plot 8: Plot an expanded view of the single-tone AM USB and message signals.

Plot 9: Plot the multi-tone AM LSB signal.

Plot 10: Plot an expanded view of the multi-tone AM LSB and message signals.

Plot 11: Plot the multi-tone AM USB signal.

Plot 12: Plot an expanded view of the multi-tone AM USB and message

B. Predict power and bandwidth for the AM SSB signals

Peak power for the AM SSB signal is calculated as before:

$$P_p = A_p^2 / 2$$

Average power for the AM SSB signal is calculated as follows:

$$P = A^2 / 2$$

An examination of the AM SSB signal plots will confirm that the signal maximum in an AM SSB signal is half of the signal maximum in its message signal.

Question 2: Predict the following values for the single-tone AM SSB signal:

peak power
average power
transmission bandwidth

C. Verify the power and bandwidth of the AM SSB signals.

M-file: Use spectral.m to generate the spectrum of the single-tone AM LSB signal.

Use spectral.m to generate the spectrum of the single-tone AM USB signal.

Plot 13: Plot the spectrum of the single-tone AM LSB signal.

Label the following values:

amplitude of each spectral component
Hz value of each spectral component
transmission bandwidth in Hz

Plot 14: Plot the spectrum of the single-tone AM USB signal.

Label the following values:

amplitude of each spectral component
Hz value of each spectral component
transmission bandwidth in Hz

M-file: To free memory, clear the variables representing the spectra of the single-tone LSB and USB signals.

Use spectral.m to generate the spectrum of the multi-tone AM LSB signal.

Use spectral.m to generate the spectrum of the multi-tone AM USB signal.

Plot 15: Plot the spectrum of the multi-tone AM LSB signal.

Label the following values:

amplitude of each spectral component
Hz value of each spectral component
transmission bandwidth in Hz

Plot 16: Plot the spectrum of the multi-tone AM USB signal.

Label the following values:

amplitude of each spectral component
Hz value of each spectral component
transmission bandwidth in Hz

M-file: To free memory, clear the variables representing the spectra of the multi-tone LSB and USB signals.

Peak power is verified in the time domain.

M-file: To calculate peak power, use the “max” command to find the single-tone AM LSB signal maximum; then square this value and divide by 2.

Calculate peak power for the single-tone AM USB signal.

Average power is verified in the frequency domain.

M-file: To calculate total average power, use `psd.m` to obtain a vector representing power spectral density of the single-tone AM LSB signal; then use the “sum” command to add the frequencies.

Calculate total average power for the single-tone AM USB signal.

Question 3: Record the values representing peak and average power for the signal-tone LSB and USB signals.

Do your calculations for bandwidth and power in Question 2 agree with the computer-generated values?

M-file: To free memory, clear the variables representing the power spectral densities of the single-tone AM LSB and USB signals.

Part 3—Recover the AM SSB signals

A. Recover the single-tone AM SSB signals

The first step in recovering an AM SSB signal is to demodulate (multiply the signal by its carrier).

When you called the function "ssb.m" you passed in the parameters setting amplitude (car_amp) and frequency (car_freq) for the carrier signal. To demodulate the AM SSB signals, you will need to reconstruct the carrier signal in the following form:

$$\text{carrier} = \text{car_amp} * (2 * \pi * \text{car_freq} * t)$$

M-file: Multiply the single-tone AM LSB signal by its carrier.
(Remember to use the "." operator.)

Use spectral.m to generate the spectrum of the demodulated single-tone AM LSB signal.

Multiply the single-tone AM USB signal by its carrier.

Use spectral.m to generate the spectrum of the demodulated single-tone AM USB signal.

To free memory, clear the variables representing the single-tone AM LSB and USB signals; and their demodulated signals.

Plot 17: Plot the spectrum of the demodulated single-tone AM LSB signal.

Plot 18: Plot the spectrum of the demodulated single-tone AM USB signal.

M-file: To free memory, clear the variables representing the spectra of the demodulated single-tone AM LSB and USB signals.

The second step in recovering an AM SSB signal is to filter the demodulated signal in order to recover the message signal.

"Lowpass" and "ideallow" are the lowpass filters available. Choose cutoff frequencies that capture only the baseband signal frequencies.

M-file: Use recoverm.m to recover and filter the baseband signal frequencies for the single-tone AM LSB signal.

Multiply the recovered signal by a factor of 4.

Use recoverm.m to recover and filter the baseband signal frequencies for the single-tone AM USB signal.

Multiply the recovered signal by a factor of 4.

To free memory, clear the variables representing the recovered single-tone AM LSB and USB signals, and their "fft" vectors.

Plot 19: Plot the recovered single-tone AM LSB and USB signals over the message signal.

B. Recover the multi-tone AM SSB signals

M-file: Multiply the multi-tone AM LSB signal by its carrier.

Use `spectral.m` to generate the spectrum of the demodulated multi-tone AM LSB signal.

Multiply the multi-tone AM USB signal by its carrier.

Use `spectral.m` to generate the spectrum of the demodulated multi-tone AM USB signal.

To free memory, clear the variables representing the multi-tone AM LSB and USB signals; and their demodulated signals.

Plot 20: Plot the spectrum of the demodulated multi-tone AM LSB signal.

Plot 21: Plot the spectrum of the demodulated multi-tone AM USB signal.

M-file: To free memory, clear the variables representing the spectra of the demodulated multi-tone AM LSB and USB signals.

Use `recoverm.m` to recover and filter the baseband signal frequencies for the multi-tone AM LSB signal.

Multiply the recovered signal by a factor of 4.

Use `recoverm.m` to recover and filter the baseband signal frequencies for the multi-tone AM USB signal.

Multiply the recovered signal by a factor of 4.

To free memory, clear the variables representing the recovered single-tone AM LSB and USB signals, and their "fft" vectors.

Plot 22: Plot the recovered multi-tone AM LSB and USB signals over the message signal.

Question 4: Is coherent detection (detection using the carrier) necessary for an AM SSB signal?

**This page is intentionally
left blank.**

Name: _____

Section: _____

EO 3513 Programming Laboratory 7 *Conventional Amplitude Modulation* (*Conventional AM*)

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

M-file and **plot** instructions refer to building MATLAB script files; **questions** can be answered separately. Develop script files, produce and label plots, and answer questions as directed by your instructor. All plots should be numbered and titled, with x- and y-axes labeled.

Part 1—Generate single- and multi-tone message signals and spectra

A. Generate the message signals

M-file: Establish a time vector with a Δt of 0.0001 and a duration of 1 second.

Generate a single-tone signal with a frequency of less than 500. Find the maximum signal value.

Generate a multi-tone signal with frequencies of less than 500. Find the maximum signal value.

Plot 1: Plot the single-tone message signal.

Plot 2: Plot the multi-tone message signal.

B. Predict power and bandwidth for the message signals

Parseval's theorem states that average signal power can be calculated in either the time or the frequency domain. The following formula applies to calculation in the time domain:

$$P = A_0^2 + \frac{1}{2} \sum_{N=1}^{\infty} (A_N^2 + B_N^2)$$

Programming Laboratory 7—page 1

Peak power is calculated as follows:

$$P_p = \frac{A_p^2}{2}$$

Question 1: Calculate the following values for the single-tone message signal:

peak power
average power
baseband bandwidth

C. Verify bandwidth for the message signals

M-file: Use spectral.m to generate the spectrum of the single-tone message signal.

Plot 3: Plot the spectrum of the single-tone signal.

Label the following values:

Hz value of the spectral component
amplitude of the spectral component
baseband signal bandwidth in Hz

M-file: Use spectral.m to generate the spectrum of the multi-tone message signal.

Plot 4: Plot the spectrum of the multi-tone signal.

Label the following values:

Hz value of the spectral components
baseband signal bandwidth in Hz

M-file: To free memory, clear the variables representing the spectra of the single- and multi-tone signals.

Part 2—Generate conventional amplitude modulated (conventional AM) signals using single- and multi-tone input

A. Generate the single- and multi-tone conventional AM signals

The function "conv_am.m" is used to normalize and modulate the message signal.

M-file: Use conv_am.m to modulate the single-tone message signal.
Use a carrier frequency of 1500 to 3000 Hz.

Use conv_am.m to modulate the multi-tone message signal,
using a different carrier frequency in the same range.

Plot 5: Plot the single-tone conventional AM signal.

Plot 6: Plot the multi-tone AM DSB signal.

B. Predict power and bandwidth for the conventional AM signals

Peak power for the single-tone conventional AM signal is calculated as follows:

$$P_p = (1 + m)^2 P_c$$

where P_c is the average power of the carrier.

Average power for the single-tone conventional AM signal is calculated as follows:

$$P = \left(1 + \frac{m^2}{2}\right) P_c$$

where P_c is the average power of the carrier.

Question 2: Predict the following values for the single-tone conventional AM signal:

peak power
average power
transmission bandwidth

C. Verify the power and bandwidth of the conventional AM signal

Amplitude of the spectral components in the sidebands of the conventional AM signal can be calculated as

$$mA/2$$

where m is the modulation index and A is the amplitude of the signal tone. The amplitude of the spectral component representing the carrier is equal to the amplitude of the carrier.

M-file: Use `spectral.m` to generate the spectrum of the single-tone conventional AM signal.

Use `spectral.m` to generate the spectrum of the multi-tone conventional AM signal.

Plot 7: Plot the spectrum of the single-tone conventional AM signal.

Label the following values:

amplitude of each spectral component
Hz value of each spectral component
transmission bandwidth in Hz

Plot 8: Plot the spectrum of the multi-tone conventional AM signal.

Label the following values:

Hz value of each spectral component
transmission bandwidth in Hz

M-file: To free memory, clear variables representing spectra of the single- and multi-tone conventional AM signals.

Peak power is verified in the time domain.

M-file: To calculate peak power, use the “max” command to find the single-tone conventional AM signal maximum; then square this value and divide by 2.

Average power is verified in the frequency domain.

M-file: To calculate total average power, use `psd.m` to obtain a vector representing power spectral density of the single-tone conventional AM signal; then use the “sum” command to add the frequencies.

To free memory, clear the variable representing power spectral density.

Question 3: Record the values representing peak and average power for the signal-tone conventional AM signal.

Do your calculations for bandwidth and power in Question 2 agree with the computer-generated values?

Part 3—Recover the conventional AM signals

A. Recover the single-tone conventional AM signal

Conventional AM signals are recovered via use of a bandpass filter and envelope detector.

The function "recoverm.m" is used to bandpass-filter and recover the signals.

M-file: Use recoverm.m and the bandpass filter idealbnd.m to filter and recover the single-tone conventional AM signal.

The function "envelope.m" is an envelope detector that uses a Hilbert transform to detect the magnitude of the complex envelope. Due to its algorithm, "envelope.m" is a highly accurate but memory-intensive function.

M-file: Reduce the size of the vector representing the filtered single-tone conventional AM signal to about 500 points.

Use envelope.m detect the filtered single-tone conventional AM signal.

Plot 9: Plot an expanded view of the single-tone envelope-detected signal over its filtered conventional AM signal.

To complete the signal detection, the steps in the conventional AM process are reversed. The DC value is removed from the envelope-detected signal; this signal is divided by the modulation index; finally the signal is multiplied by the maximum value of the original message signal.

M-file: Subtract 1 from the envelope-detected signal to remove the DC value. Then divide by the modulation index and multiply by the maximum value in the message signal.

Plot 10: Plot an expanded view of the single-tone message signal over the modified envelope-detected signal.

B. Recover the multi-tone conventional AM signal

M-file: Use `recoverm.m` and the bandpass filter `idealbnd.m` to filter and recover the multi-tone conventional AM signal.

Reduce the size of the vector representing the filtered multi-tone conventional AM signal to about 500 points.

Use `envelope.m` to detect the filtered multi-tone conventional AM signal.

Plot 11: Plot an expanded view of the multi-tone envelope-detected signal over its filtered conventional AM signal.

M-file: Subtract 1 from the envelope-detected signal to remove the DC value. Then divide by the modulation index and multiply by the maximum value in the message signal.

Plot 12: Plot an expanded view of the multi-tone message signal over the modified envelope-detected signal.

Part 4—Observe the effect of overmodulation on a conventional AM signal

A. Overmodulate a single-tone conventional AM signal

M-file: Use `conv_am.m` to modulate the single-tone signal, using a modulation index greater than 1.

Using the envelope detector on the conventional AM signal will give you a preview of why envelope detection is inadequate as a detection method for an overmodulated conventional AM signal.

M-file: Reduce the length of the overmodulated conventional AM signal to about 500 points.

Use `envelope.m` to detect the envelope of the overmodulated signal.

Plot 13: Plot an expanded view of the envelope-detected signal over the overmodulated conventional AM signal.

B. Describe the effect of overmodulation on signal recovery

Question 4: What type of detection is needed for an overmodulated conventional AM signal? Why?

**This page is intentionally
left blank.**

Name: _____

Section: _____

EO 3513 Programming Laboratory 8

Frequency Modulation (FM)

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

M-file and **plot** instructions refer to building MATLAB script files; **questions** can be answered separately. Develop script files, produce and label plots, and answer questions as directed by your instructor. All plots should be numbered and titled, with x- and y-axes labeled.

Part 1--Observe the FM modulation process for single- and multi-tone input

A. Generate the message signals

M-file: Establish a time vector with a Δt of 0.0001 and a duration of 1 second.

Generate a single-tone signal with a frequency of less than 100.

Generate a two-tone signal with frequencies of less than 100. Amplitudes may vary.

Plot 1: Plot the single-tone message signal.

Plot 2: Plot the multi-tone message signal.

B. Predict peak power, average power, and bandwidth for the message signals

Parseval's theorem states that average signal power can be calculated in either the time or the frequency domain. The following formula applies to calculation in the time domain:

$$P = A_0^2 + \frac{1}{2} \sum_{N=1}^{\infty} (A_N^2 + B_N^2)$$

Peak power is calculated as follows:

$$P_P = \frac{A_P^2}{2}$$

Question 1: Calculate the following values for the single-tone message signal:

**peak power
average power
baseband bandwidth**

C. Generate the single- and multi-tone message signals spectra

M-file: Use spectral.m to generate the spectrum of the single-tone message signal.

Plot 3: Plot the spectrum of the single-tone message signal.

Label the following values for the message signal:

**Hz value of the spectral component
amplitude of the spectral component
baseband signal bandwidth in Hz**

M-file: Use spectral.m to generate the spectrum of the multi-tone message signal.

Plot 4: Plot the spectrum of the multi-tone message signal.

Label the following values for the message signal:

**Hz value of the spectral components
amplitude of the spectral components
baseband signal bandwidth in Hz**

M-file: To free memory, clear the variables associated with the single- and multi-tone message signal spectra.

Part 2--Generate frequency modulated (FM) signals using single- and multi-tone input

A. Generate the single- and multi-tone FM signals

The function "fm_mod.m" is used to frequency modulate the message signal. Input parameters relating to the message signal frequency and phase, and carrier signal frequency and amplitude must be specified. Additionally, either beta (β) or delta_f (Δf) must be specified. When β is specified, Δf is calculated and returned, and vice versa.

For multi-tone FM signals, signal frequencies and phases are passed in using vectors. For this laboratory, phase (theta) can be set to zero. A sample frequency vector follows:

`fm=[33 66]`

M-file: Use fm_mod.m to generate the single-tone FM signal.

Use fm_mod.m to generate the multi-tone FM signal.

For both FM signals, use $\beta = 10$ and a carrier frequency between 1500 and 2500 Hz. Set the amplitude of each FM signal equal to the maximum amplitude in its message signal.

Plot 5: Plot the single-tone FM signal over its message signal. Notice the variations in FM signal frequency as message signal amplitude changes.

Plot 6: Plot the multi-tone FM signal over its message signal. Show the message signal maximum and minimum amplitudes, if possible.

B. Predict peak power, average power, and bandwidth for the FM signals

Peak power is calculated as before:

$$P_P = \frac{A_P^2}{2}$$

Average power of the FM signal is calculated as follows:

$$P = A^2 / 2$$

Recall that B and Δf are related in that $B f_m = \Delta f$. There are three cases for estimating transmission bandwidth, depending on the value of B :

for $B \leq 0.25$	$B_T \approx 2 f_m$	(narrowband FM)
for $2.5 \leq B < 10$	$B_T \approx 2 (1 + B) f_m$	(Carson's rule)
for $B \geq 10$	$B_T \approx 2 B f_m$	(wideband FM)

where f_m is the frequency of the message signal.

Question 2: Predict the following values for the single-tone FM signal:

peak power
average power
maximum frequency deviation Δf
transmission bandwidth (use Carson's rule)

C. Generate the spectra of the FM signals

M-file: Use `spectral.m` to generate the spectrum of the single-tone FM signal.

Plot 7: Plot the spectrum of the single-tone FM signal.

Label the carrier frequency and the transmission bandwidth.

Label Δf to each side of the carrier frequency.

Question 3: Consult a table of values for Bessel functions (or use the MATLAB "bessel" command). How many sidebands are required for 98% power transmission for this FM signal? Does the spectrum shown in Plot 7 reflect the expected number of sidebands?

Question 4: What is the distance between the sidebands in the FM spectrum shown in Plot 7?

M-file: Use `spectral.m` to generate the spectrum of the multi-tone FM signal.

When calculating transmission bandwidth for a multi-tone FM signal, use the maximum frequency in the message signal.

Plot 8: Plot the spectrum of the multi-tone FM signal.

Label the carrier frequency and transmission bandwidth.

D. Verify the power and bandwidth of the single-tone FM signal

Peak power is verified in the time domain.

M-file: To calculate peak power, square the maximum value of the FM signal and divide by 2.

Average power is verified in the frequency domain.

M-file: To calculate total average power, use `psd.m` to obtain a vector representing power spectral density of the single-tone conventional AM signal; then use the "sum" command to add the frequencies.

Question 5: Record the values representing peak and average power for the signal-tone FM signal.

Do your calculations for bandwidth and power in Question 2 agree with the computer-generated values?

Part 3--Control the bandwidth of the FM signals

A. Control the bandwidth of the single-tone FM signal by varying β

M-file: Use `fm_mod.m` to generate two single-tone FM signals, both at a carrier frequency of 2500 Hz but with different values of β .

Question 6: Calculate the maximum frequency deviation Δf associated with each of the two values of β .

Question 7: Calculate the transmission bandwidth for each of the single-tone FM signals.

M-file: Use `spectral.m` to generate the spectra of the two single-tone FM signals.

Plot 9: Plot the first of the single-tone FM signal spectra generated above.

Label the spectrum with the transmission bandwidth calculated in Question 7.

Plot 10: Plot the second of the single-tone FM signal spectra generated above.

Label the spectrum with the transmission bandwidth calculated in Question 7.

A. Control the bandwidth of the multi-tone FM signal by varying Δt

M-file: Use `fm_mod.m` to generate two multi-tone FM signals, both at a carrier frequency of 2500 Hz but with different values of Δt .

Question 8: Calculate the value of B associated with each of the two values of Δf .

Question 9: Calculate the transmission bandwidth for each of the multi-tone FM signals (use the higher of the two values for message signal frequency).

M-file: Use `spectral.m` to generate the spectra of the two multi-tone FM signals.

Plot 11: Plot the first of the multi-tone FM signal spectra generated above.

Label the spectrum with the transmission bandwidth calculated in Question 9.

Plot 12: Plot the second of the single-tone FM signal spectra generated above.

Label the spectrum with the transmission bandwidth calculated in Question 9.

Name: _____

Section: _____

EO 3513 Programming Laboratory 9

Radio Frequency Digital Modulation Methods (ASK, FSK, BPSK, and QPSK)

This laboratory requires the Communications Toolbox for use with MATLAB. The toolbox functions are available on disk for use on both PC and Macintosh platforms. Users should employ the "help" feature in MATLAB for information about the functions, or consult the users' guide.

M-file and **plot** instructions refer to building MATLAB script files; **questions** can be answered separately. Develop script files, produce and label plots, and answer questions as directed by your instructor. All plots should be numbered and titled, with x- and y-axes labeled.

Part 1—Amplitude shift keying (ASK)

A. Generate the digital message signal

The radio frequency modulation methods in this laboratory are all based on digital message signals. One method of generating a random bitstream in MATLAB is described below:

```
bitstream=round(rand(1:500));   %random bitstream
```

M-file: **Generate a random bitstream of 100 bits.**

ASK signals require a unipolar digital signal.

M-file: **Use "nrzluni.m" to generate a digital message signal with a bit rate of 100 bits per second.**

Question 1: **Calculate the bit duration τ for this signal.**

Plot 1: **Plot at least the first 10 bits of the NRZL unipolar digital message signal.**

Label the values (0 or 1) of the first 10 bits, and the bit duration τ .

Recall that a "coarse" approximation for baseband bandwidth of a digital signal is $0.5/\tau$.

Question 2: Calculate the approximate baseband bandwidth of the NRZL unipolar digital message signal.

M-file: Use "spectral.m" to generate the one-sided spectrum of the digital message signal.

Plot 2: Plot the first 2000 Hz of the NRZL unipolar digital message signal spectrum.

Label the baseband bandwidth in Hz.

B. Generate the ASK signal

M-file: Generate the ASK signal by modulating it with a cosine with a carrier frequency of between 500 and 1000 Hz. (Remember to use the ".*" operator.)

Plot 3: Plot at least 5 bits of the ASK signal.

Label the values (0 or 1) of the bits shown.

Question 3: Why is ASK modulation often referred to as "on-off keying"?

M-file: Use "spectral.m" to generate the one-sided spectrum of the ASK signal.

Plot 4: Plot the first 2000 Hz of the ASK signal spectrum.

Label the carrier frequency of the ASK signal.

M-file: To free memory, clear the ASK signal and spectrum, and the NRZL unipolar digital message spectrum.

Part 2—Frequency shift keying (FSK)

A. Generate the FSK signal

The FSK signal can be based on either a unipolar or bipolar digital signal. Use the NRZL unipolar digital message signal generated in Part 1 to generate the FSK signal.

The FSK signal is generated using the function "fsk.m." Within this function, the bits representing 1's are modulated at a higher frequency, and the bits representing 0's are modulated at a lower frequency.

M-file: Use "fsk.m" to generate the FSK signal. Choose a high frequency of between 1000 and 1500 and a low frequency of between 500 and 1000, allowing a margin of 200 Hz between the two frequencies.

Plot 5: Plot at least the first 5 bits of the FSK signal over the NRZL unipolar digital message signal.

Label the values (0 or 1) of the bits shown.

M-file: Use "spectral.m" to generate the one-sided spectrum of the FSK signal.

Plot 6: Plot the first 2000 Hz of the FSK signal spectrum.

Label the two carrier frequencies of the FSK signal.

M-file: To free memory, clear the FSK signal and spectrum and the NRZL unipolar digital message signal.

Part 3—Binary phase shift keying (BPSK)

A. Generate the digital message signal

The BPSK signal is based on a bipolar digital signal.

M-file: Set the bit pattern for the first 8 bits of the bitstream as follows:

`bitstream(1:8)=[0 0 0 1 1 0 1 1];`

Use "nrzlb.m" to generate an NRZL bipolar digital message signal at a bit rate of 100 bits per second.

Plot 7: Plot at least the first 10 bits of the NRZL bipolar digital message signal.

Label the values (0 or 1) of the first 10 bits, and the bit duration τ .

M-file: Use "spectral.m" to generate the spectrum of the NRZL bipolar digital message signal.

Plot 8: Plot the first 2000 Hz of the spectrum of the NRZL bipolar digital message signal.

Label the baseband bandwidth in Hz.

B. Generate the BPSK signal

M-file: Generate the BPSK signal by modulating the message signal with a cosine at a frequency of between 250 and 1000 Hz.

Plot 9: Plot at least the first 5 bits of the NRZL bipolar digital message signal over the BPSK signal. Notice the phase shifts present in the signal where bit values change.

Label the values (0 or 1) of the bits shown.

B. Generate the BPSK spectrum

M-file: Use "spectral.m" to generate the one-sided spectrum of the BPSK signal.

Plot 10: Plot the first 2000 Hz of the BPSK signal spectrum.

Label the carrier frequency of the BPSK signal.

Part 4—Quadrature phase shift keying (QPSK)

The QPSK signal, like the BPSK signal, must be based on a bipolar digital message signal; however, it exhibits four different types of phase shifts. The first 8 bits of the digital message signal form a pattern that will demonstrate the four types of phase shifts present in the QPSK signal.

A. Generate the QPSK signal

The first step in generating the QPSK signal is to split the signal by putting it through a serial-to-parallel converter. One of the output signals is composed of the odd bits, the other of the even bits. The bits in each output signal have a bit rate of half of the input signal.

M-file: Use "ser_par.m" to split the NRZL bipolar digital message signal into two signals.

Plot 11: Plot at least the first 5 bits of the signal composed of odd bits.

Label the values (0 or 1) of the first 5 bits shown, and the bit duration τ .

Plot 12: Plot at least the first 5 bits of the signal composed of even bits.

Label the values (0 or 1) of the first 5 bits shown, and the bit duration τ .

Next, the signal composed of odd bits must be modulated by a positive cosine function. The signal composed of even bits must be modulated by a negative sine function.

M-file: Multiply the signal composed of odd bits with a positive cosine function with a carrier frequency of between 250 and 1000 Hz.

Plot 13: Plot at least the first 5 bits of the digital message signal over the modulated "odd-bit" signal. Notice the phase shifts present.

M-file: Using the same carrier frequency, multiply the signal composed of even bits with a negative sine function.

Plot 14: Plot at least the first 5 bits of the digital message signal over the modulated "even-bit" signal. Notice the phase shifts present.

The QPSK modulation process is completed by summing the two modulated signals in the time domain.

M-file: Sum the two modulated signals.

Plot 15: Plot the first 5 bits of the QPSK signal (0.1 seconds).

Label the phase shifts present in the signal.

M-file: Use "spectral.m" to generate the one-sided QPSK spectrum.

Plot 16: Plot the first 2000 Hz of the QPSK spectrum.

Label the carrier frequency of the QPSK signal.

Question 4: Compare the spectrum for the BPSK signal in Plot 10 with that of the QPSK signal in Plot 16. What is the chief advantage of quadriphase shift keying over bipolar phase shift keying?

**This page is intentionally
left blank.**

APPENDIX D—PROGRAMMING LABORATORY KEYS

EO 3513 Programming Laboratory 1 Key *Signal and Spectrum Generation*

Answers will vary. The answers below are based on the following signals:

$s1=5*\cos(2*\pi*30*t1);$

$s2=10*\cos(2*\pi*100*t2);$

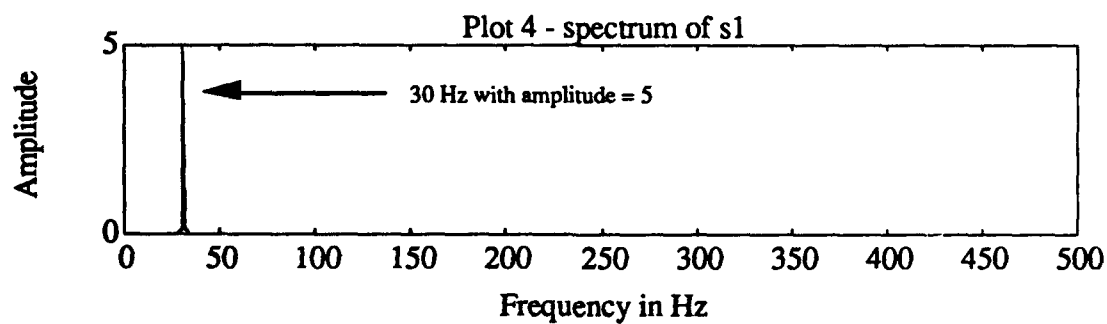
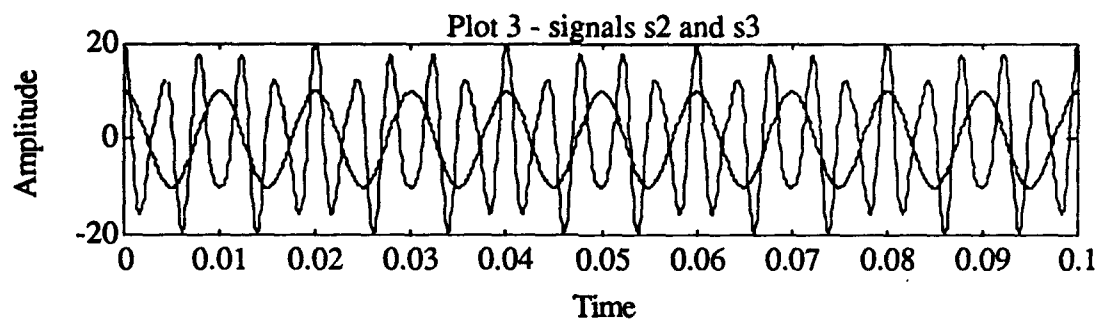
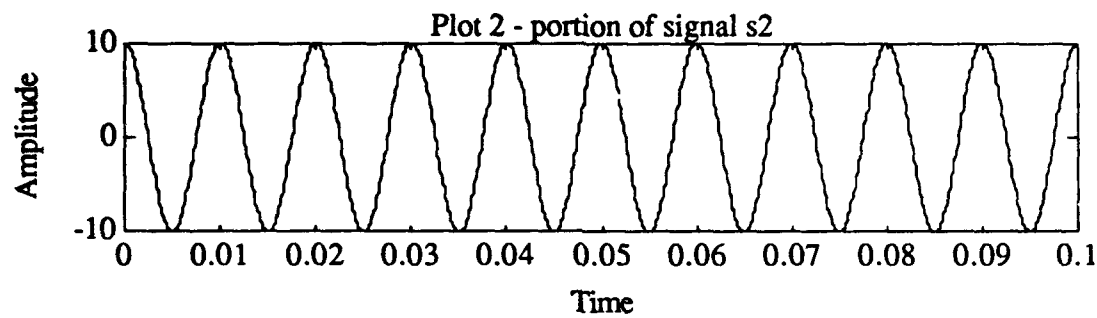
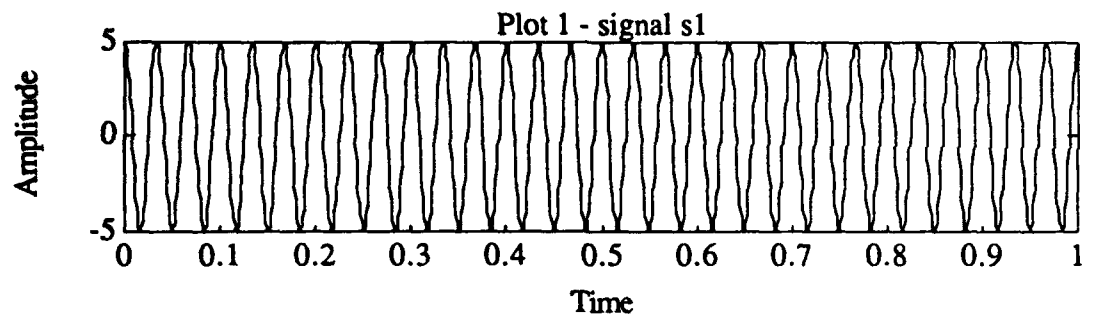
$s3=15*\cos(2*\pi*250*t2)+5*\cos(2*\pi*400*t2);$

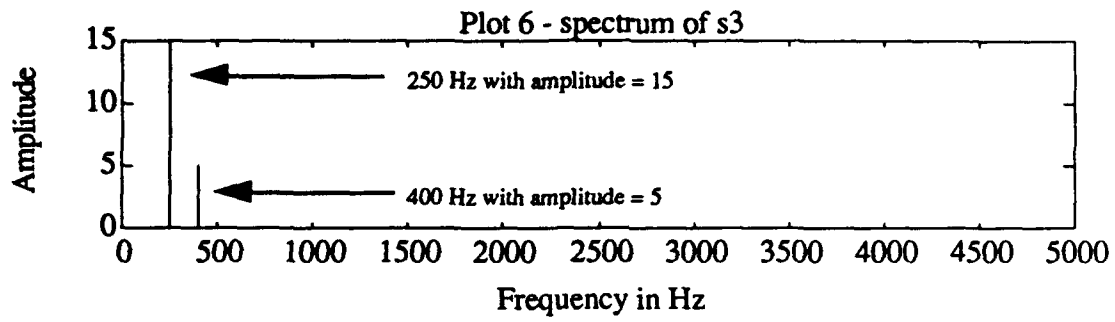
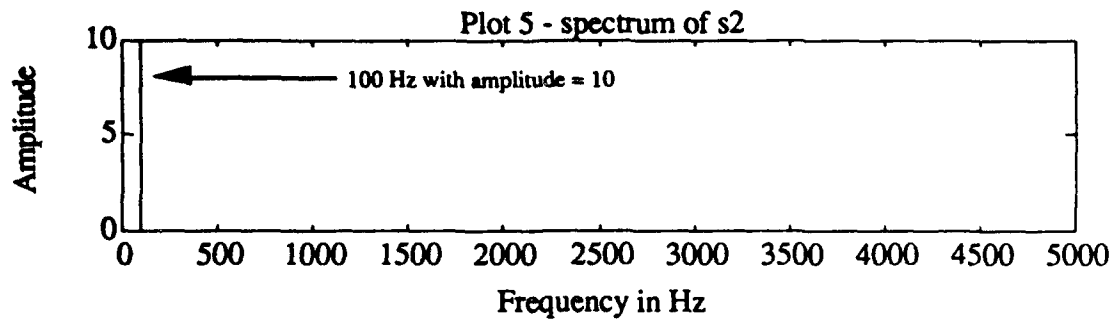
Question 1: Compare Plots 4 and 5. Why does Plot 5 display more frequencies than Plot 4?

Answer: For signal s1 shown in Plot 4, the step size of 0.001 produced a one-sided spectrum of only 500 Hz; for signal s2 shown in Plot 5, the step size of 0.0001 produced a one-sided spectrum of 5000 Hz.

Question 2: Given the spectral plot of a single-tone signal, how could you determine the maximum amplitude and frequency for the signal?

Answer: On a one-sided spectrum, signal amplitude is plotted against frequency in Hz. The maximum signal amplitude could be found by observing the amplitude of the spectral component. Frequency could be determined by observing the Hz value of the spectral component.





lab1_ex.m

%Programming Lab 1 example for instructor use
%Answers will vary!

%%
%Programming Lab 1 Signal and Spectrum Generation
%%
%Part 1--Produce and plot signals
%A. Establish a time vector

clear
clg

t1=0:.001:1;
t2=0:.0001:1;

%B. Generate a signal

s1=5*cos(2*pi*30*t1);
s2=10*cos(2*pi*100*t2);
s3=15*cos(2*pi*250*t2)+5*cos(2*pi*400*t2);

%C. Controlling signal plots

%Plot 1

subplot(211),
plot(t1,s1)
title('Plot 1 - signal s1')
xlabel('Time')
ylabel('Amplitude')

%Plot 2

plot(t2(1:1000),s2(1:1000))
title('Plot 2 - portion of signal s2')
xlabel('Time')
ylabel('Amplitude')

pause
clg

%Plot 3

subplot(211),
plot(t2(1:1000),s3(1:1000))
title('Plot 3 - signals s2 and s3')

```

xlabel('Time')
ylabel('Amplitude')
hold on
plot(t2(1:1000),s2(1:1000),'b')
hold off

%%%%%%%%%%%%%%
%Part 2--Produce and plot spectra
%A. Calling a function

[spec1,shortHz]=spectral(s1,.001); %generate spectrum for s1

%Plot 4

plot(shortHz,spec1) %plot spectrum of s1
title('Plot 4 - spectrum of s1')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clc

[spec2,longHz]=spectral(s2,.0001); %generate spectrum for s2

%Plot 5

subplot(211),
plot(longHz,spec2) %plot spectrum of s2
title('Plot 5 - spectrum of s2')
xlabel('Frequency in Hz')
ylabel('Amplitude')

[spec3]=spectral(s3,.0001); %generate spectrum for s3

%Plot 6

plot(longHz,spec3) %plot spectrum of s3
title('Plot 6 - spectrum of s3')
xlabel('Frequency in Hz')
ylabel('Amplitude')

```

**This page is intentionally
left blank.**

EO 3513 Programming Laboratory 2A Key

Natural Sampling and Recovery

Answers will vary. The answers below are based on the following signal:

$$s=2*(\cos(2*\pi*150*t)+\cos(2*\pi*250*t)+\cos(2*\pi*450*t));$$

Question 1: What is the maximum amplitude of the signal?

Answer: 6

Question 2: What is the highest frequency in the signal? What is the Nyquist rate?

Answer: highest frequency is 450 Hz
Nyquist rate is 900 Hz

Question 3: Calculate the following values for the sampled signal:

sampling period T (calculated in seconds)
pulse duration τ (calculated in seconds)

Describe the pulse shape of the sampled signal.

Answer: $T = 1/f_s \Rightarrow 1/1000 \Rightarrow 0.001$ seconds
 $\tau = d * T \Rightarrow 0.5 * 0.001 \Rightarrow 0.0005$ seconds

The top of each pulse reflects the shape of the message signal.

Question 4: Calculate P_N for $N = 1$, $N = 2$, and $N = 3$.

Compare with the values shown on the spectral plot.

Answer: For $N = 1$ $P_N = 0.3183$
For $N = 2$ $P_N = 0$
For $N = 3$ $P_N = -0.1061$

Values on the spectral plot appear larger by a factor of 2, consistent with the increased signal amplitude. (Note that absolute values are plotted.)

Question 5: Describe the overall shape of the spectrum. Does the spectrum conform to your theoretical expectations? Note any discrepancies.

Answer: The spectrum consists of groups of frequencies which have a "sinc" shape to their envelope.

Yes--no discrepancies.

Question 6: Compare Plot 5 with Plot 2. What is the effect of undersampling on the spectrum?

Answer: The replicas of the baseband message signal frequencies produced by sampling overlap.

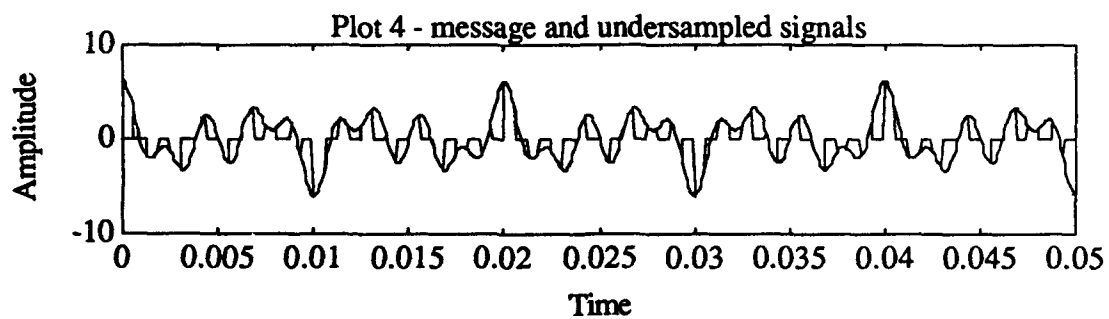
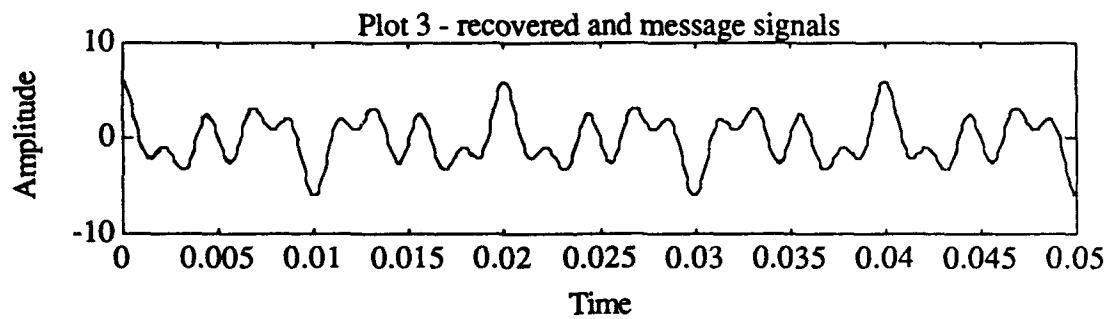
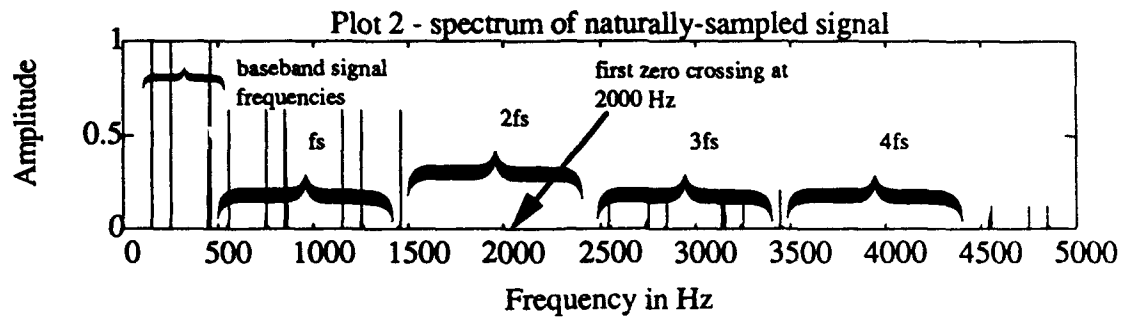
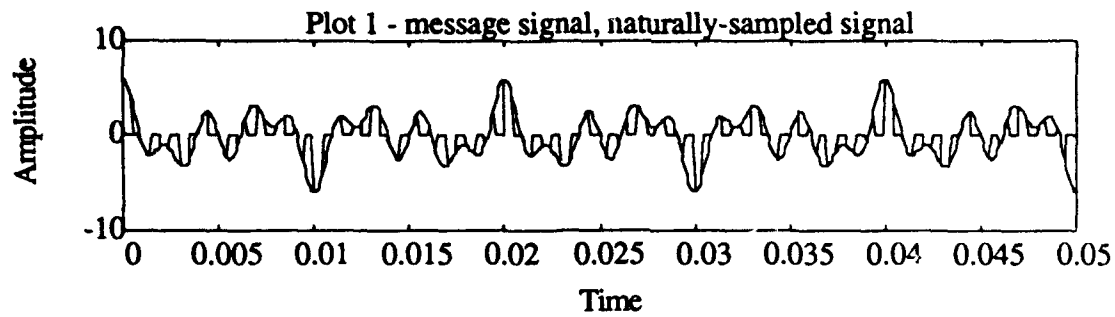
Question 7: What is the effect of undersampling on signal recovery?

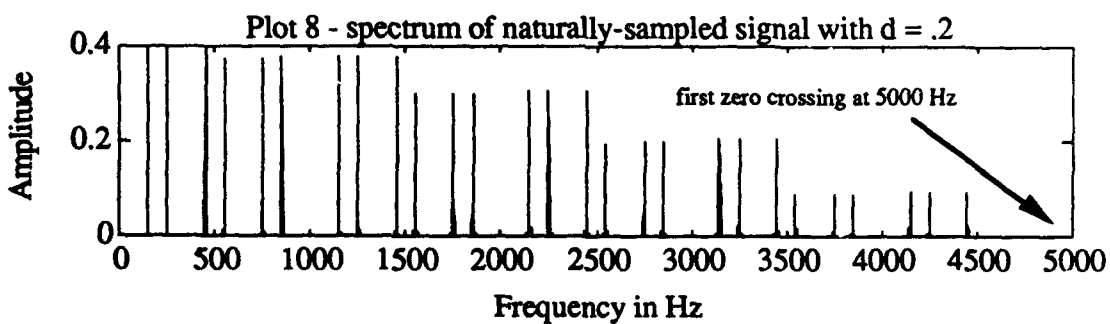
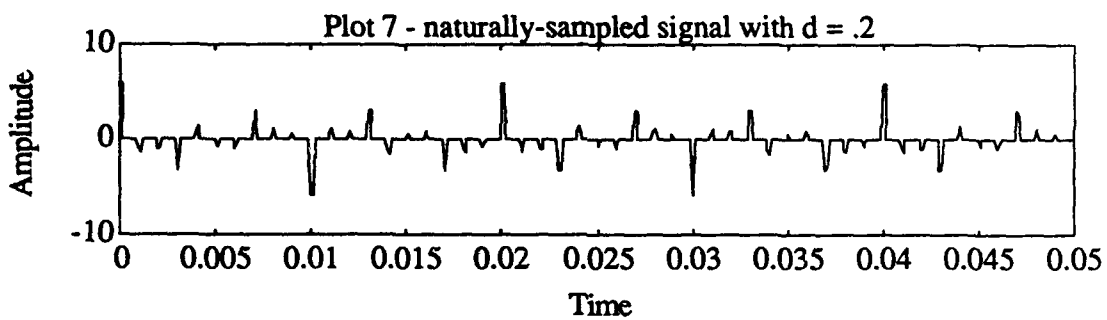
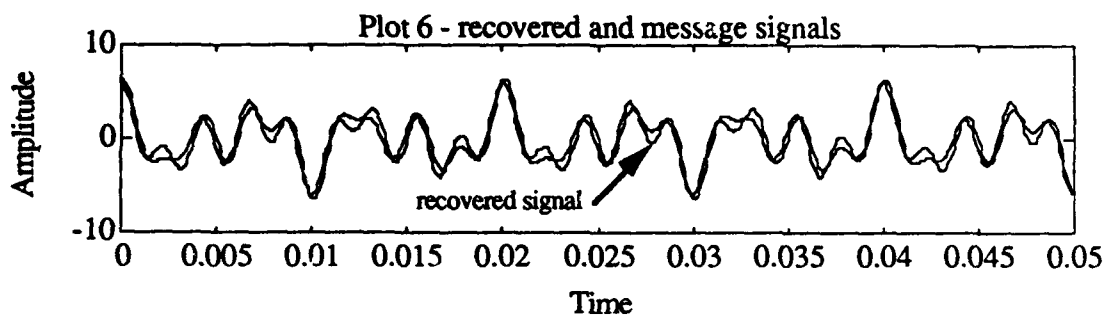
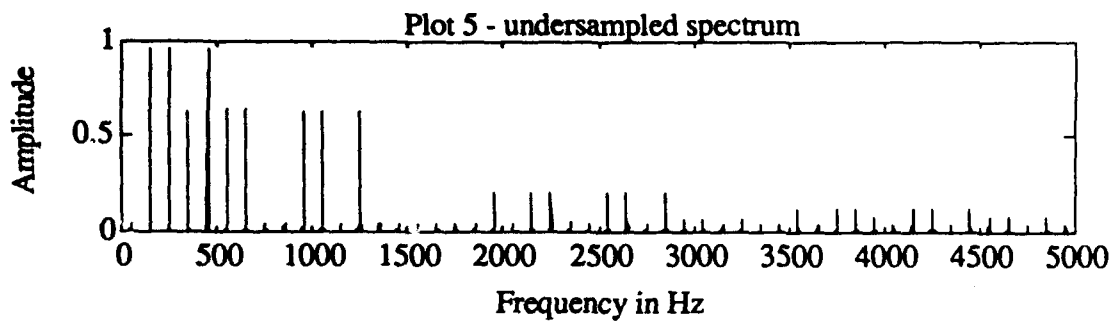
Answer: The overlapping of the baseband message signal frequencies prevents proper recovery of the message signal (this effect is called "aliasing").

Question 8: Compare Plot 8 with Plot 2. What is the effect of changing the duty cycle on the sampled signal baseband bandwidth?

Answer: As the pulse width decreases, the sampled signal baseband bandwidth increases; as pulse width increases, the sampled signal baseband bandwidth decreases. This relationship illustrates the trade-off between transmission power and bandwidth requirements.

Narrower pulses translate to more frequent changes, requiring higher frequencies to capture those changes.





lab2A_ex.m

%Programming Lab 2A example for instructor use
%Answers will vary!

%%
%Programming Lab 2A Natural Sampling and Recovery
%%
%Part 1--Generate a naturally-sampled signal and its spectrum
%A. Generate a signal

clear
clg

delta_t=.0001;
samprate=1000;

t=0:delta_t:1; %generate the signal
s=2*(cos(2*pi*150*t)+cos(2*pi*250*t)+cos(2*pi*450*t)); %signal

%Plot 1

subplot(211), %plot the signal
plot(t(1:500),s(1:500))
title('Plot 1 - message signal, naturally-sampled signal')
xlabel('Time')
ylabel('Amplitude')
hold on

%B. Naturally-sample the signal

natsig1=natsamp(s,delta_t,samprate,.5); %sample the signal

plot(t(1:500),natsig1(1:500),'b') %plot the naturally-sampled signal
hold off

%C. Generate the spectrum

[specnat1,Hz,fftnat1]=spectral(natsig1,delta_t);

%Plot 2

subplot(212), %plot the spectrum
plot(Hz,specnat1)
title('Plot 2 - spectrum of naturally-sampled signal')
xlabel('Frequency in Hz')
ylabel('Amplitude')

```

pause
clg

%%%%%%%%%%%%%%
%Part 2--Recover the message signal
%A. Filter and recover the sampled signal

recsig1=recover(fftmat1,..5,'ideal',Hz,500);

%Plot 3

subplot(211), %plot recovered signal
plot(t(1:500),[recsig1(1:500);s(1:500)])
title('Plot 3 - recovered and message signals')
xlabel('Time')
ylabel('Amplitude')

clear %free memory

%%%%%%%%%%%%%%
%Part 3--Observe the effects of aliasing
%A. Produce an undersampled signal

delta_t=.0001; %restore variables for t and s
samprate=800;

t=0:delta_t:1;
%s=5*cos(2*pi*200*t)+8*cos(2*pi*300*t)+3*cos(2*pi*450*t);
s=2*(cos(2*pi*150*t)+cos(2*pi*250*t)+cos(2*pi*450*t)); %signal

natsig2=natsamp(s,delta_t,samprate,.5); %undersample the signal

%Plot 4

subplot(212), %plot undersampled signal over message signal
plot(t(1:500),[s(1:500);natsig2(1:500)])
title('Plot 4 - message and undersampled signals')
xlabel('Time')
ylabel('Amplitude')

pause
clg

[specmat2,Hz,fftmat2]=spectral(natsig2,delta_t);

%Plot 5

subplot(211), %plot undersampled spectrum

```

```

plot(Hz,specnat2)
title('Plot 5 - undersampled spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

recsig2=recover(fftmat2,.5,'ideallow',Hz,500);

%Plot 6

subplot(212), %plot recovered undersampled signal
plot(t(1:500),[recsig2(1:500);s(1:500)])
title('Plot 6 - recovered and message signals')
xlabel('Time')
ylabel('Amplitude')

pause
clg
clear

%%%%%%%%%%%%%%
%Part 4--Observe the effect on the spectrum of varying the duty cycle
%A. Generate the sampled signal

delta_t=.0001;
samprate=1000;

t=0:delta_t:1; %generate the signal
s=2*(cos(2*pi*150*t)+cos(2*pi*250*t)+cos(2*pi*450*t)); %signal

d=.2;

natsig3=natsamp(s,delta_t,samprate,d); %sample the signal

%Plot 7

subplot(211), %plot the naturally-sampled signal
plot(t(1:500),natsig3(1:500))
title('Plot 7 - naturally-sampled signal with d = .2')
xlabel('Time')
ylabel('Amplitude')

%B. Generate the spectrum

[specnat3,HZ]=spectral(natsig3,delta_t);

%Plot 8

subplot(212), %plot the spectrum

```

```
plot(Hz,specnat3)
title('Plot 8 - spectrum of naturally-sampled signal with d = .2')
xlabel('Frequency in Hz')
ylabel('Amplitude')
```

EO 3513 Programming Laboratory 2B Key

Flattop Sampling and Recovery

Answers will vary. The answers below are based on the following signal:

$$s = \cos(2\pi \cdot 100 \cdot t) + \cos(2\pi \cdot 150 \cdot t) + \cos(2\pi \cdot 200 \cdot t);$$

Question 1: What is the maximum amplitude of the signal?

Answer: 3

Question 2: What is the highest frequency in the signal? What is the Nyquist rate?

Answer: highest frequency is 200 Hz
Nyquist rate is 400 Hz

Question 3: Calculate the following values for the sampled signal:

sampling period T (calculated in seconds)
pulse duration τ (calculated in seconds)

Describe the pulse shape of the sampled signal.

Answer: $T = 1/f_s \Rightarrow 1/1000 \Rightarrow 0.001$ seconds
 $\tau = d \cdot T \Rightarrow 0.4 \cdot 0.001 \Rightarrow 0.0004$ seconds

The top of each pulse is flat, reflecting the amplitude of the message signal at the pulse beginning.

Question 4: Calculate P_n for $f = 550$, $f = 700$, and $f = 880$.

Compare with the values shown on the spectral plot.

Answer: For $f = 800$ $P_N = 0.3359$
For $f = 850$ $P_N = 0.3282$
For $f = 900$ $P_N = 0.3200$

Values are consistent with those on the spectral plots.

Question 5: Describe the overall shape of the spectrum. Does the spectrum conform to your theoretical expectations? Note any discrepancies.

Answer: The flattop-sampled signal spectrum shows frequencies with amplitudes that individually conform to the "sinc" envelope.

Yes--no discrepancies.

Question 6: Compare Plot 5 with Plot 2. What is the effect of undersampling on the spectrum?

Answer: The replicas of the baseband message signal frequencies produced by sampling overlap.

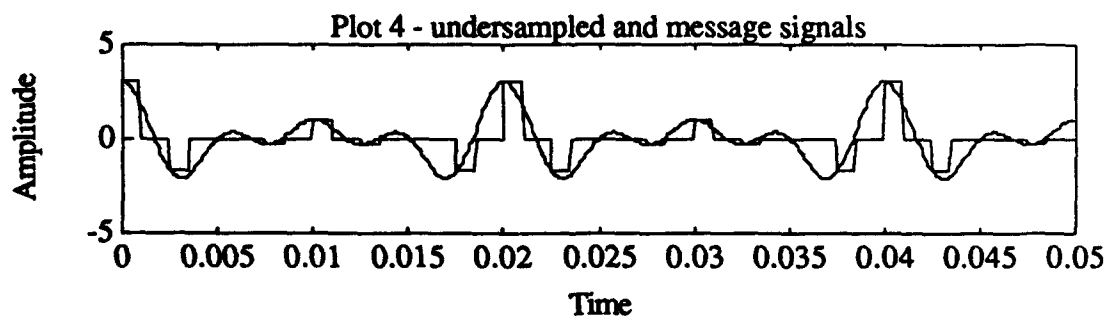
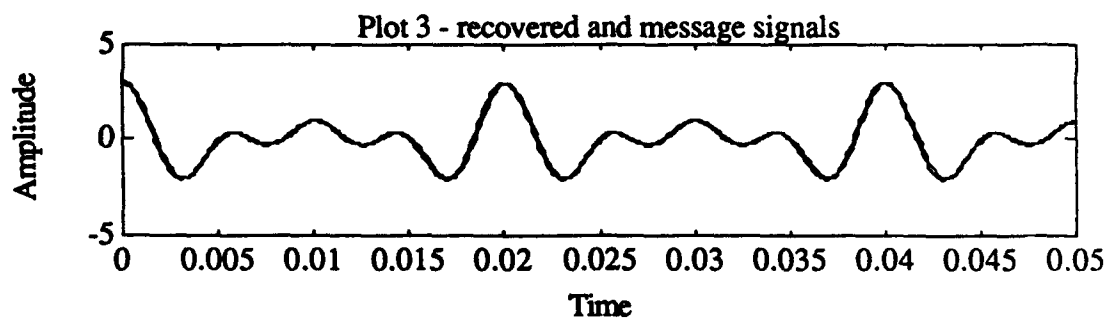
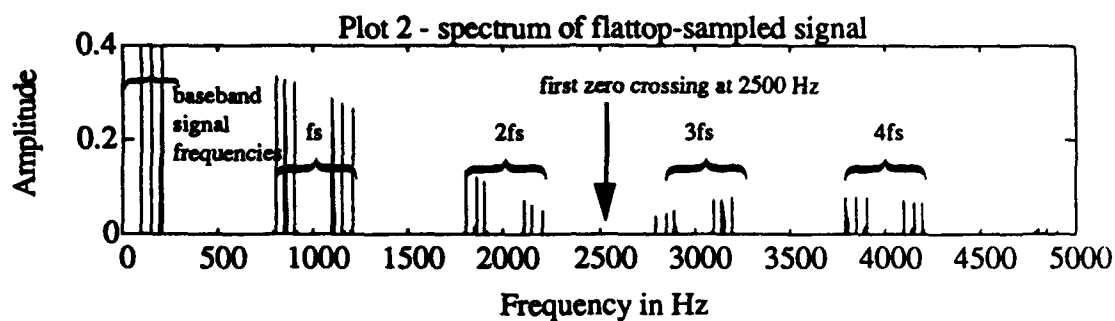
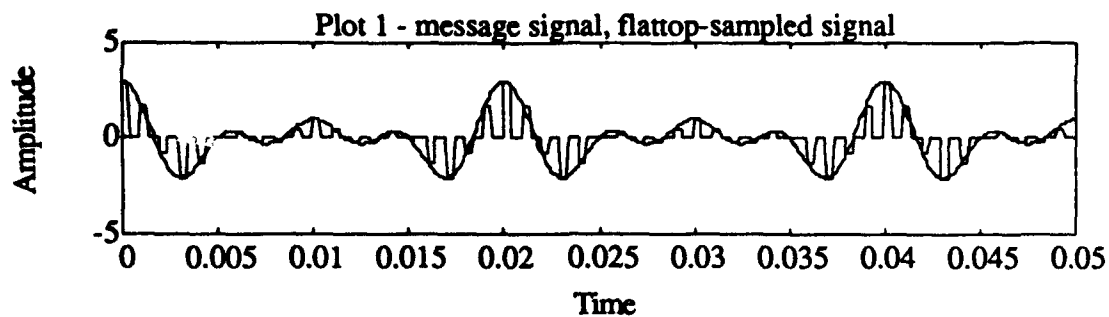
Question 7: What is the effect of undersampling on signal recovery?

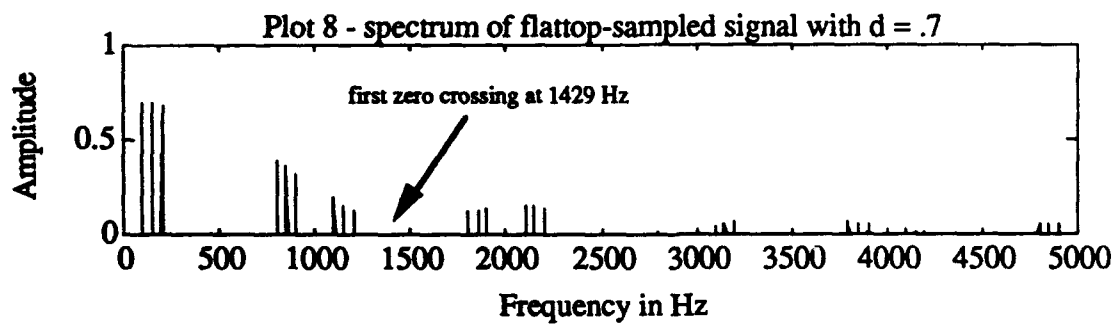
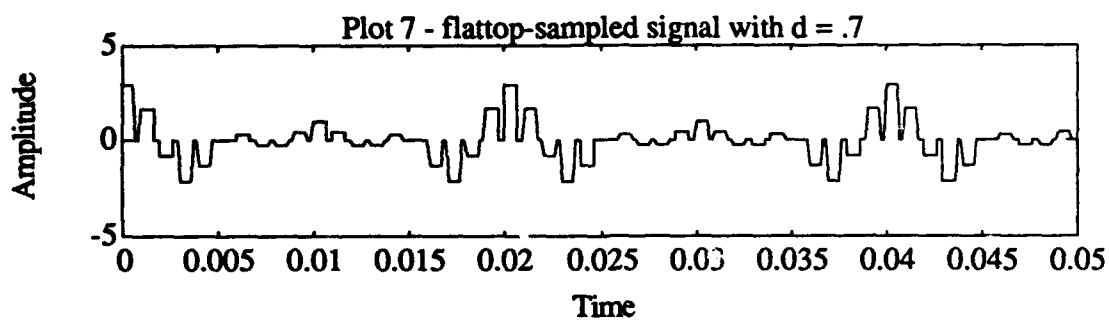
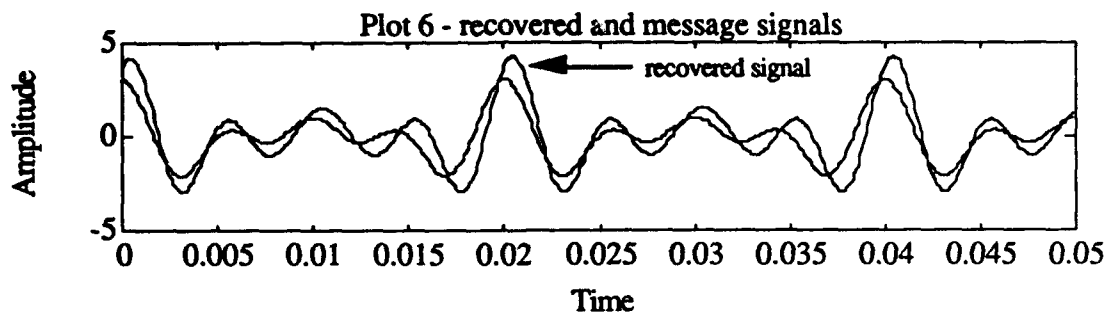
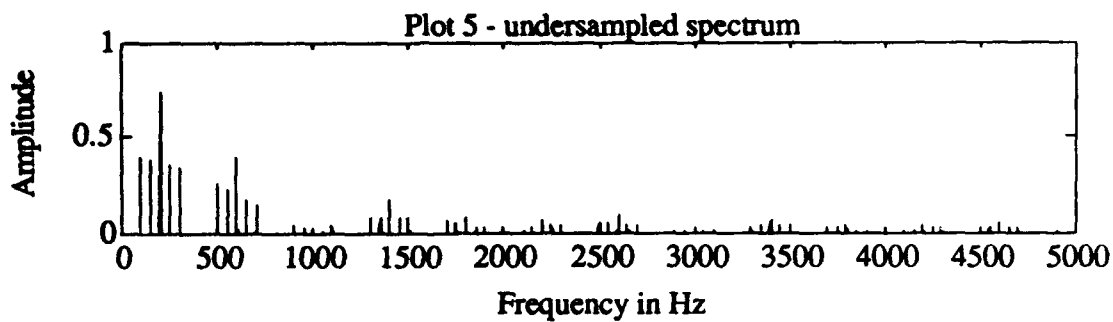
Answer: The overlapping of the baseband message signal frequencies prevents proper recovery of the message signal (this effect is called "aliasing").

Question 8: Compare Plot 8 with Plot 2. What is the effect of changing the duty cycle on the sampled signal baseband bandwidth?

Answer: As the pulse width decreases, the sampled signal baseband bandwidth increases; as pulse width increases, the sampled signal baseband bandwidth decreases. This relationship illustrates the trade-off between transmission power and bandwidth requirements.

Narrower pulses translate to more frequent changes, requiring higher frequencies to capture those changes.





lab2B_ex.m

%Programming Lab 2B example for instructor use

%Answers will vary!

%%

%Programming Lab 2B Flattop Sampling and Recovery

%%

%Part 1--Generate a flattop-sampled signal and its spectrum

%A. Generate a signal

clear

clg

delta_t=.0001;

samprate=1000;

t=0:delta_t:1; %generate the signal

s=cos(2*pi*100*t)+cos(2*pi*150*t)+cos(2*pi*200*t);

%Plot 1

subplot(211), %plot the signal

plot(t(1:500),s(1:500))

title('Plot 1 - message signal, flattop-sampled signal')

xlabel('Time')

ylabel('Amplitude')

hold on

%B. Flattop-sample the signal

flatsig1=flattop(s,delta_t,samprate,.4); %sample the signal

%Plot 2

%plot the flattop-sampled signal over

plot(t(1:500),flatsig1(1:500),'b') %the message signal

hold off

%C. Generate the spectrum

[specflat1,HZ,fftflat1]=spectral(flatsig1,delta_t);

%Plot 3

subplot(212), %plot the spectrum

plot(HZ,specflat1)

title('Plot 2 - spectrum of flattop-sampled signal')

```

xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

%%%%%%%%%%%%%%
%Part 2--Recover the message signal
%A. Filter and recover the sampled signal

recsig1=recover(ffflat1,4,'ideallow',Hz,250);

%Plot 3

subplot(211), %plot recovered signal
plot(t(1:500),[recsig1(1:500);s(1:500)])
title('Plot 3 - recovered and message signals')
xlabel('Time')
ylabel('Amplitude')

clear %free memory

%%%%%%%%%%%%%%
%Part 3--Observe the effects of aliasing
%A. Produce an undersampled signal

delta_t=.0001; %restore variables for t and s
samprate=400;

t=0:delta_t:1;
s=cos(2*pi*100*t)+cos(2*pi*150*t)+cos(2*pi*200*t);

flatsig2=flattop(s,delta_t,samprate,.4); %undersample the signal

%Plot 4

subplot(212), %plot undersampled signal and message signal
plot(t(1:500),[flatsig2(1:500);s(1:500)])
title('Plot 4 - undersampled and message signals')
xlabel('Time')
ylabel('Amplitude')

pause
clg

[specflat2,Hz,ffflat2]=spectral(flatsig2,delta_t);

%Plot 5

```

```

subplot(211), %plot undersampled spectrum
plot(Hz,specflat2)
title('Plot 5 - undersampled spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

recsig2=recover(fftflat2,4,'ideallow',Hz,250);

%Plot 6

subplot(212), %plot recovered undersampled signal
plot(t(1:500),[recsig2(1:500);s(1:500)])
title('Plot 6 - recovered and message signals')
xlabel('Time')
ylabel('Amplitude')

pause
clg
clear

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Part 4--Observe the effect on the spectrum of varying the duty cycle
%A. Generate the sampled signal

delta_t=.0001;
samprate=1000;

t=0:delta_t:1; %generate the signal
s=cos(2*pi*100*t)+cos(2*pi*150*t)+cos(2*pi*200*t); %signal

d=.7;

flatsig3=flattop(s,delta_t,samprate,d); %sample the signal

%Plot 7

subplot(211), %plot the flattop-sampled signal
plot(t(1:500),flatsig3(1:500))
title('Plot 7 - flattop-sampled signal with d = .7')
xlabel('Time')
ylabel('Amplitude')

%B. Generate the spectrum

[specflat3,HZ]=spectral(flatsig3,delta_t);

%Plot 8

```

```
subplot(212), %plot the spectrum
plot(Hz,specflat3)
title('Plot 8 - spectrum of flattop-sampled signal with d = .7')
xlabel('Frequency in Hz')
ylabel('Amplitude')
```

EO 3513 Programming Laboratory 2C Key

Impulse Sampling and Recovery

Answers will vary. The answers below are based on the following signal:
 $s=5*(\cos(2*\pi*100*t)+\cos(2*\pi*350*t)+\cos(2*\pi*400*t));$

Question 1: What is the maximum amplitude of the signal?

Answer: 15

Question 2: What is the highest frequency in the signal? What is the Nyquist rate?

Answer: highest frequency is 400 Hz
Nyquist rate is 800 Hz

Question 3: Calculate the duration of the sampling period T (in seconds).

Answer: $T = 1/f_s \Rightarrow 1/2000 \Rightarrow 0.0005$ seconds

Question 4: Describe the overall shape of the spectrum. Does the spectrum conform to your theoretical expectations? Note any discrepancies.

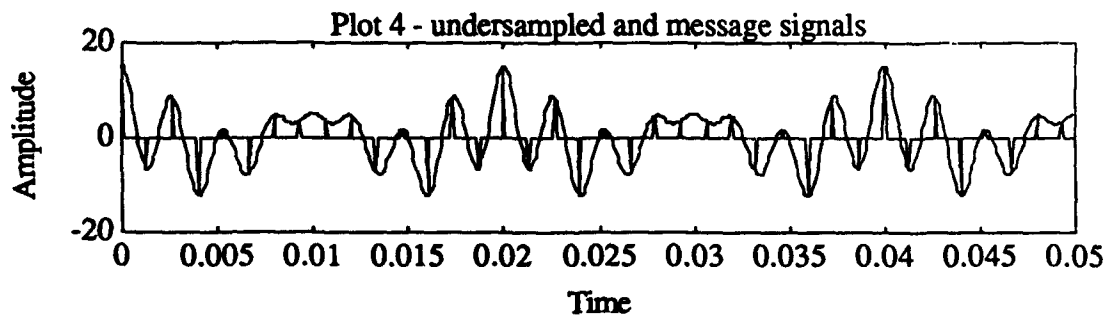
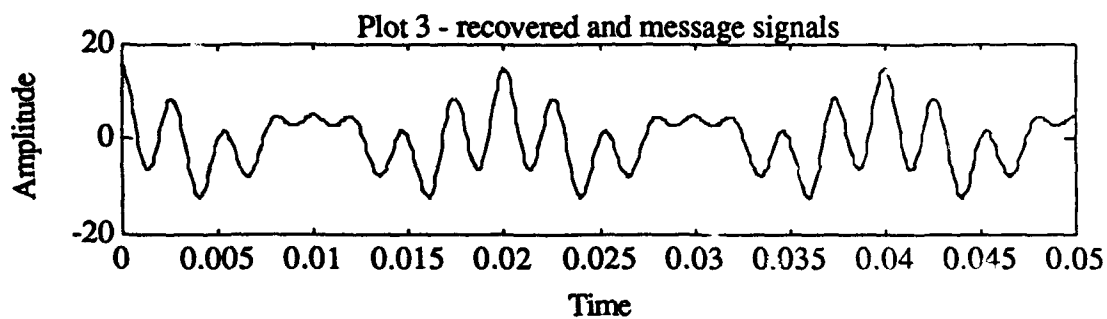
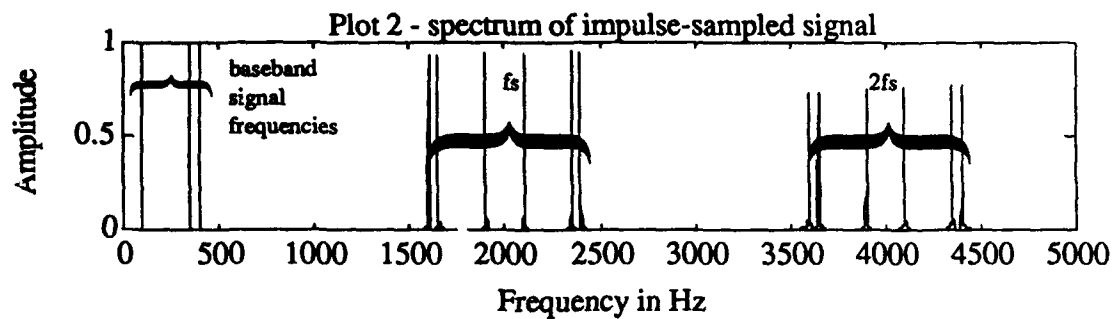
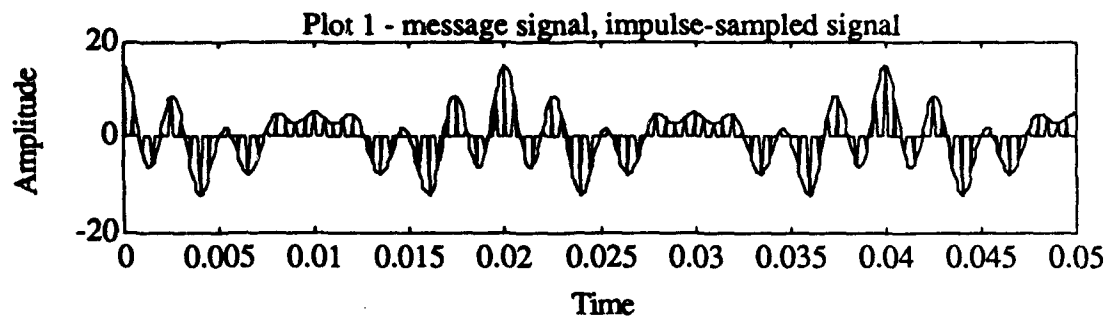
Answer: The amplitudes of the spectral components of an impulse-sampled spectrum should in theory remain constant. Since those shown in the plot decline gradually (due to the computer's inability to generate a perfect impulse), the spectrum does not conform to theoretical expectations.

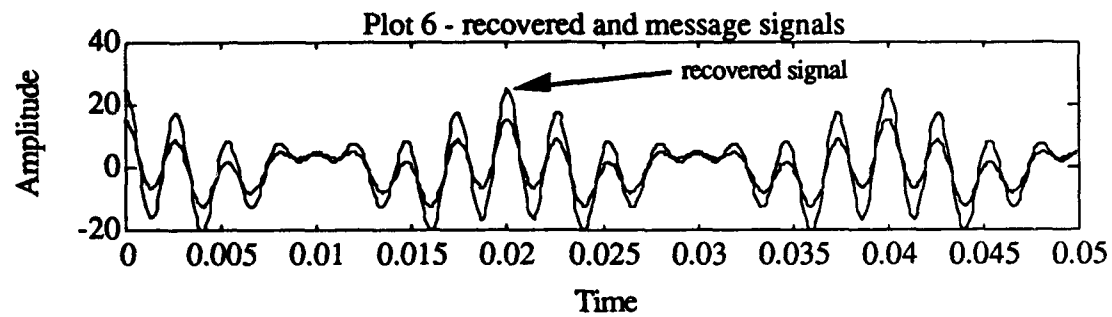
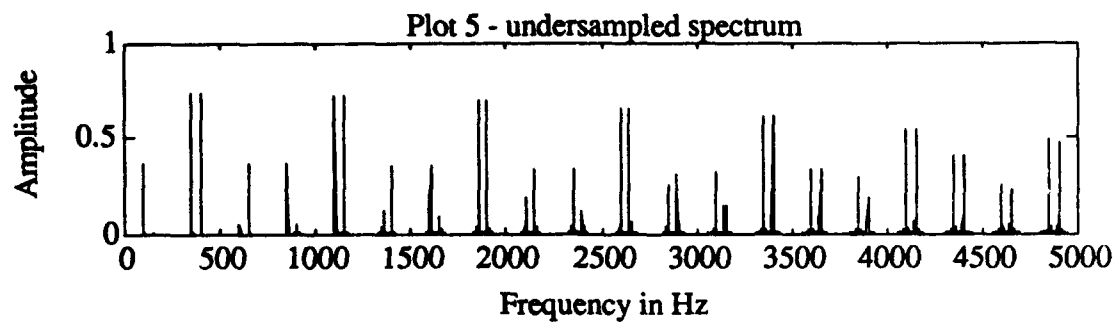
Question 5: Compare Plot 5 to Plot 2. What is the effect of undersampling on the spectrum?

Answer: The replicas of the baseband message signal frequencies produced by sampling overlap.

Question 6: What is the effect of undersampling on signal recovery?

Answer: The overlapping of the baseband message signal frequencies prevents proper recovery of the message signal (this effect is called "aliasing").





lab2C_ex.m

```
%Programming Lab 2C example for instructor use
%Answers will vary!

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Programming LabC Impulse Sampling and Recovery
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Part 1--Generate an impulse-sampled signal and its spectrum
%A. Generate a signal

clear
clg

delta_t=.0001;
samprate=2000;
d=samprate*delta_t;

t=0:delta_t:1; %generate the signal
s=5*(cos(2*pi*100*t)+cos(2*pi*350*t)+cos(2*pi*400*t));

%Plot 1

subplot(211), %plot the signal
plot(t(1:500),s(1:500))
title('Plot 1 - message signal, impulse-sampled signal')
xlabel('Time')
ylabel('Amplitude')
hold on

%B. Impulse-sample the signal

impsig1=impsamp(s,delta_t,samprate); %sample the signal

%plot the impulse-sampled signal over
plot(t(1:500),impsig1(1:500),'g') %the message signal
hold off

%C. Generate the spectrum

[specimp1,H,ftimp1]=spectral(impsig1,delta_t);

%Plot 2

subplot(212), %plot the spectrum
plot(H,specimp1)
title('Plot 2 - spectrum of impulse-sampled signal')
xlabel('Frequency in Hz')
```

```

ylabel('Amplitude')

pause
clg

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Part 2--Recover the impulse-sampled signal
%A. Filter and recover the sampled signal

recsig1=recovers(ffimp1,d,'ideallow',Hz,500);

%Plot 3

subplot(211), %plot recovered signal
plot(t(1:500),[recsig1(1:500);s(1:500)])
title('Plot 3 - recovered and message signals')
xlabel('Time')
ylabel('Amplitude')

clear %free memory

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Part 3--Observe the effects of aliasing
%A. Produce an undersampled signal

delta_t=.0001; %restore variables for t and s
samprate=750;
d=samprate*delta_t;

t=0:delta_t:1;
s=5*(cos(2*pi*100*t)+cos(2*pi*350*t)+cos(2*pi*400*t));

impsig2=impsamp(s,delta_t,samprate); %undersample the signal

%Plot 4

subplot(212), %plot undersampled signal
plot(t(1:500),[impsig2(1:500);s(1:500)])
title('Plot 4 - undersampled and message signals')
xlabel('Time')
ylabel('Amplitude')

pause
clg

[specimp2,Hz,ffimp2]=spectral(impsig2,delta_t);

%Plot 5

```

```

subplot(211), %plot undersampled spectrum
plot(Hz,specimp2)
title('Plot 5 - undersampled spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

recsig2=recovers(ffimp2,d,'ideallow',Hz,500);

%Plot 6

subplot(212), %plot recovered undersampled signal
plot(t(1:500),[recsig2(1:500);s(1:500)])
title('Plot 6 - recovered and message signals')
xlabel('Time')
ylabel('Amplitude')

```

EO 3513 Programming Laboratory 3A Key Pulse Modulation (PAM and PWM)

Answers will vary. The answers below are based on the following signal:
 $s=2*(\cos(2*\pi*120*t)+\sin(2*\pi*30*t));$

Question 1: What is the maximum amplitude of the message signal?

Answer: 4

Question 2: Calculate the following values, in seconds, for the PAM signal:

sampling period T
pulse duration τ

Answer: $T = 1/f_s \Rightarrow 1/500 \Rightarrow 0.002$ seconds
 $\tau = d * T \Rightarrow 0.5 * .002 = 0.001$ seconds

Question 3: What is the maximum pulse duration that could occur in your PWM signal?

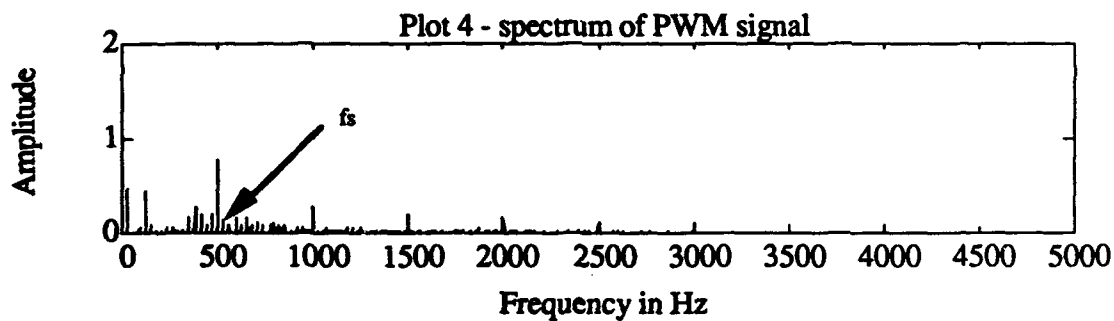
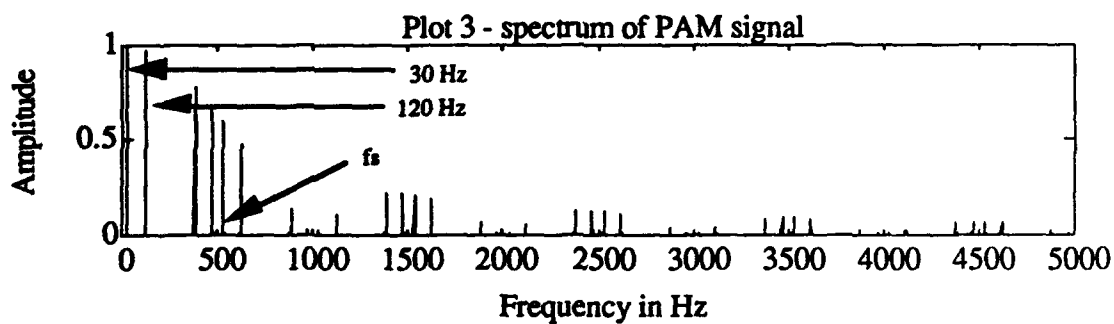
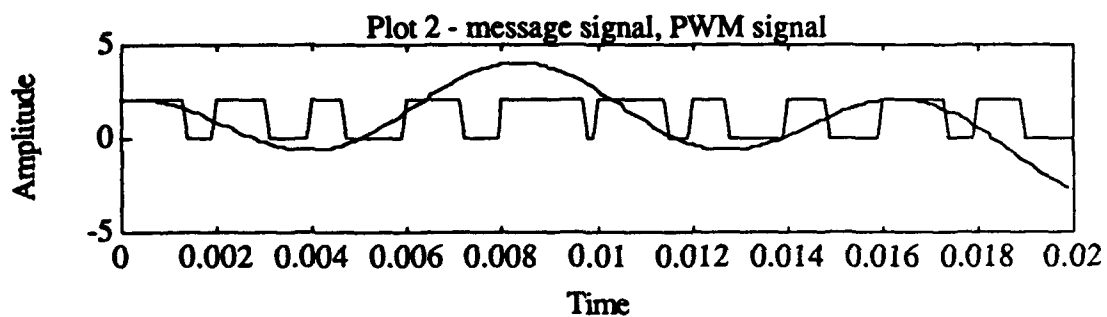
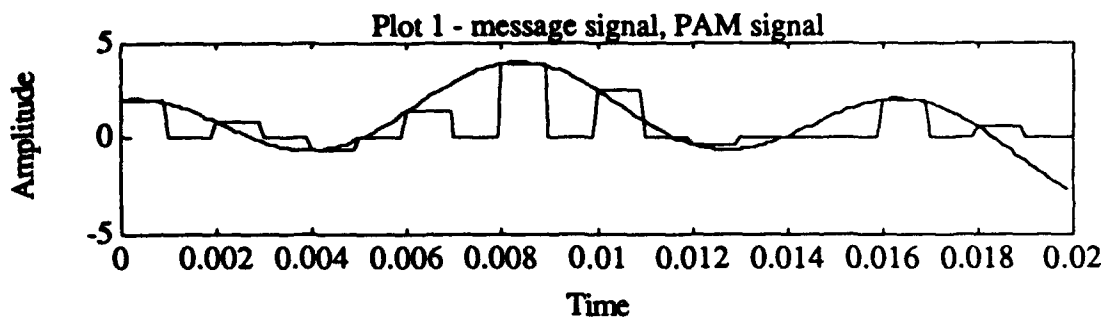
Answer: Maximum pulse duration = $0.9 * T \Rightarrow 0.9 * 0.002 \Rightarrow 0.0018$ seconds

Question 4: Using the above approximations, calculate the baseband bandwidths for the PAM and PWM signals. Do these values reflect what you observe in the spectral plots? Note any discrepancies.

Answer: PAM bandwidth = $0.5/\tau \Rightarrow 0.5/0.001 \Rightarrow 500$ Hz
PWM bandwidth = $0.5/\text{risetime} \Rightarrow 0.5/0.0001 = 5000$ Hz

The calculated baseband bandwidth of the PAM signal, 500 Hz, is adequate to capture the signal information. The PAM pulses occur at fixed, known intervals, and are of a fixed, known duration. The 500 Hz approximation is based solely on τ , the value of that duration.

The PWM signal requires a much higher baseband bandwidth because less information is known about its pulses. The higher frequencies are needed to convey the information regarding the exact widths of the pulses. The approximation of 5000 Hz appears to capture most of the information required for the PWM signal.



lab3A_ex.m

%Programming Lab 3A example for instructor use
%Answers will vary!

%%
%Programming Lab 3A Pulse Amplitude and Pulse Width Modulation
%%
%Part 1--Observe the differences in the time domain
%for two types of pulse-modulated signals (PAM and PWM)
%A. Generate a signal

clg
clear

delta_t=.0001;
t=0:delta_t:1;
s=2*(cos(2*pi*120*t)+sin(2*pi*30*t));
samprate=500;

%B. Generate the PAM and PWM signals

flatsig=flattop(s,delta_t,samprate,.5);

%Plot 1

subplot(211),
plot(t(1:200),[flatsig(1:200);s(1:200)])
title('Plot 1 - message signal, PAM signal')
xlabel('Time')
ylabel('Amplitude')

pwsig=pulswid(s,delta_t,samprate,.9); %pulse-width modulate
%the signal

%Plot 2

subplot(212),
plot(t(1:200),[pwsig(1:200);s(1:200)])
title('Plot 2 - message signal, PWM signal')
xlabel('Time')
ylabel('Amplitude')

pause
clg

%%
%Part 2--Observe the differences in the spectra
%for two types of pulse-modulated signals (PAM and PWM)

%A. Generate the spectrum of the PAM signal

```
[specpam,HZ]=spectral(flatsig,delta_t); %generate the PAM spectrum
```

%Plot 3

```
subplot(211),  
plot(HZ,specpam)  
title('Plot 3 - spectrum of PAM signal')  
xlabel('Frequency in Hz')  
ylabel('Amplitude')
```

%B. Generate the spectrum of the PWM signal

```
specpwm=spectral(pwsig,delta_t); %generate the PWM spectrum
```

%Plot 4

```
subplot(212),  
plot(HZ,specpwm)  
title('Plot 4 - spectrum of PWM signal')  
xlabel('Frequency in Hz')  
ylabel('Amplitude')
```

EO 3513 Programming Laboratory 3B Key *Pulse Modulation (PAM and PPM)*

Answers will vary. The answers below are based on the following signal:

$$s=5*(\cos(2*\pi*100*t)+\cos(2*\pi*40*t));$$

Question 1: What is the maximum amplitude of the message signal?

Answer: 10

Question 2: Calculate the following values, in seconds, for the PAM signal:

sampling period T
pulse duration τ

Answer: $T = 1/f_s \Rightarrow 1/500 \Rightarrow 0.002$ seconds
 $\tau = d * T \Rightarrow 0.5 * .002 = 0.001$ seconds

Question 3: What is the largest pulse offset that could occur in your PPM signal?

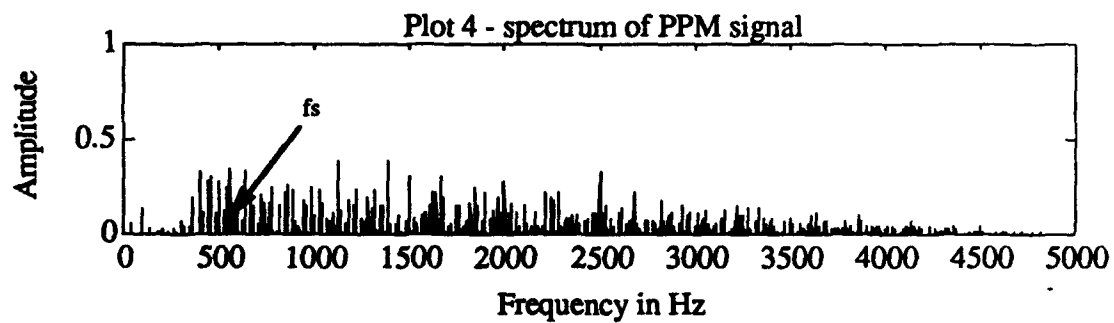
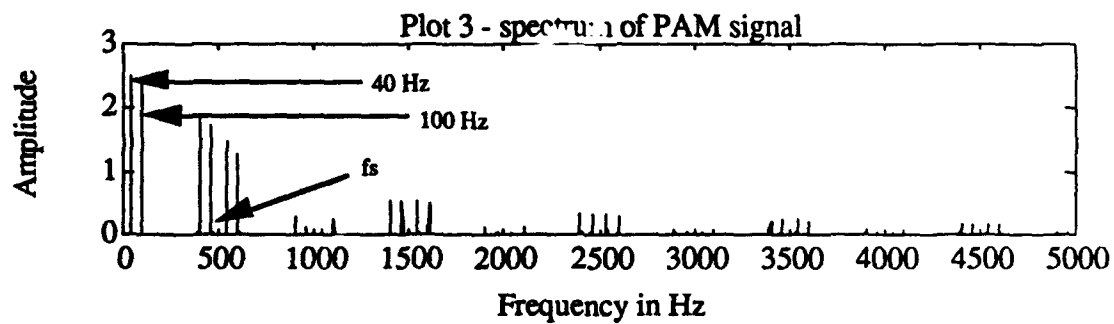
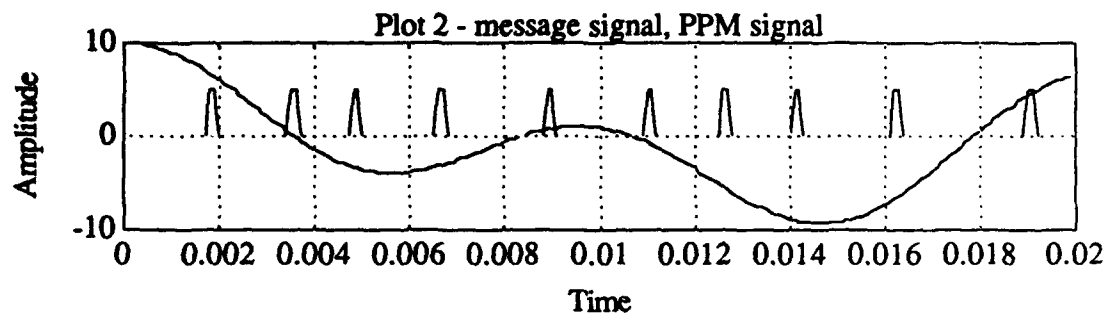
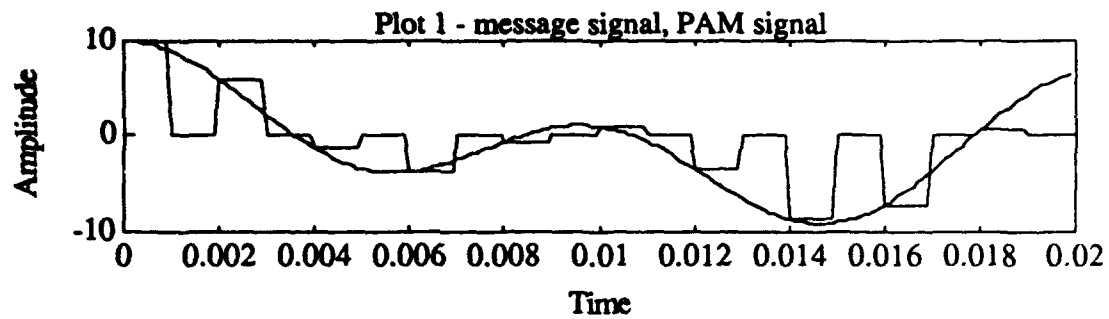
Answer: Largest pulse offset = $0.9 * T \Rightarrow 0.9 * 0.002 \Rightarrow 0.0018$ seconds

Question 4: Using the above approximations, calculate the baseband bandwidths for the PAM and PPM signals. Do these values reflect what you observe in the spectral plots? Note any discrepancies.

Answer: PAM bandwidth = $0.5/\tau \Rightarrow 0.5/0.001 \Rightarrow 500$ Hz
PPM bandwidth = $0.5/\text{risetime} \Rightarrow 0.5/0.0001 = 5000$ Hz

The calculated baseband bandwidth of the PAM signal, 500 Hz, is adequate to capture the signal information. The PAM pulses occur at fixed, known intervals, and are of a fixed, known duration. The 500 Hz approximation is based solely on τ , the value of that duration.

The PPM signal requires a much higher baseband bandwidth because less information is known about its pulses. The higher frequencies are needed to convey the information regarding the exact location of the pulses. The approximation of 5000 Hz appears to capture all most of the information required for the PPM signal.



lab3B_ex.m

%Programming Lab 3B example for instructor use
%Answers will vary!

%%
%Programming Lab 3B Pulse Amplitude and Pulse Position Modulation
%%
%Part 1--Observe the differences in the time domain
%for two types of pulse-modulated signals (PAM and PPM)
%A. Generate a signal

clg
clear

delta_t=.0001;
t=0:delta_t:1;
s=5*(cos(2*pi*100*t)+cos(2*pi*40*t));
samprate=500;

%B. Generate the PAM and PPM signals

flatsig=flattop(s,delta_t,samprate,.5);

%Plot 1

subplot(211),
plot(t(1:200),[flatsig(1:200);s(1:200)])
title('Plot 1 - message signal, PAM signal')
xlabel('Time')
ylabel('Amplitude')

ppsig=pulspos(s,delta_t,samprate,.1); %pulse-position modulate
%the signal

%Plot 2

subplot(212),
plot(t(1:200),[ppsig(1:200);s(1:200)])
title('Plot 2 - message signal, PPM signal')
xlabel('Time')
ylabel('Amplitude')
grid

pause
clg

%%
%Part 2--Observe the differences in the spectra

%for two types of pulse-modulated signals (PAM and PPM)

%A. Generate the spectrum of the PAM signal

[specpam,HZ]=spectral(flatsig,delta_t); %generate the PAM spectrum

%Plot 3

subplot(211),

plot(HZ,specpam)

title('Plot 3 - spectrum of PAM signal')

xlabel('Frequency in Hz')

ylabel('Amplitude')

%B. Generate the spectrum of the PPM signal

specppm=spectral(ppsig,delta_t); %generate the PPM spectrum

%Plot 4

subplot(212),

plot(HZ,specppm)

title('Plot 4 - spectrum of PPM signal')

xlabel('Frequency in Hz')

ylabel('Amplitude')

EO 3513 Programming Laboratory 4A Key *Analog-to-Digital Conversion (Quantization)*

Answers will vary. The answers below are based on the following signal:

$$s=4*\cos(2*\pi*20*t)+5*\cos(2*\pi*45*t);$$

Question 1: Calculate the following values relating to the quantization characteristic for the 3-bit bipolar converter:

dynamic range
actual step size
actual resolution
percentage resolution
number of levels

Answer: dynamic range = $6 * \text{number of bits} \Rightarrow 6 * 3 \Rightarrow 18 \text{ dB}$
actual step size = $2^{-n+1} \times \text{full-scale V} \Rightarrow \pm 2^{-3+1} \times 10 \Rightarrow 2.5 \text{ V}$
actual resolution = $\pm 2^{-n} \times \text{full-scale V} \Rightarrow \pm 2^{-3} \times 10 \Rightarrow 1.25 \text{ V}$
percentage resolution = $\pm 2^{-n} \times 100\% \Rightarrow \pm 2^{-3} \times 100\% \Rightarrow 12.5\%$
number of levels = 3 bits $\Rightarrow 2^3 \Rightarrow 8 \text{ levels}$

Question 2: Calculate the following values relating to the quantization characteristic for the 5-bit unipolar converter:

dynamic range
actual step size
actual resolution
percentage resolution
number of levels

Answer: dynamic range = $6 * \text{number of bits} \Rightarrow 6 * 3 \Rightarrow 18 \text{ dB}$
actual step size = $2^{-n+1} \times \text{full-scale V} \Rightarrow \pm 2^{-6+1} \times 10 \Rightarrow 0.3125 \text{ V}$
actual resolution = $\pm 2^{-(n+1)} \times \text{full-scale V} \Rightarrow \pm 2^{-6} \times 10 \Rightarrow 0.156 \text{ V}$
percentage resolution = $\pm 2^{-(n+1)} \times 100\% \Rightarrow \pm 2^{-6} \times 100\% \Rightarrow 1.56\%$
number of levels = 3 bits $\Rightarrow 2^5 \Rightarrow 32 \text{ levels}$

Question 3: List the amplitude ("voltage") of the 3-bit bipolar quantized signal in each of the first 5 sampling periods.

Answer: 7.5 V
7.5 V
7.5 V
5.0 V
5.0 V

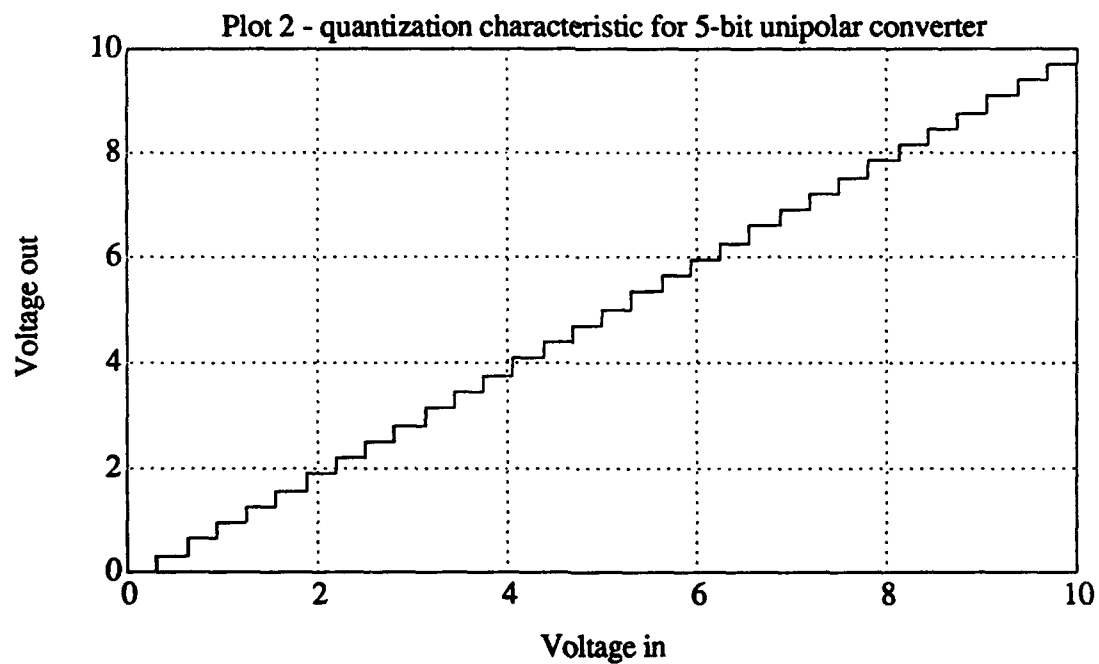
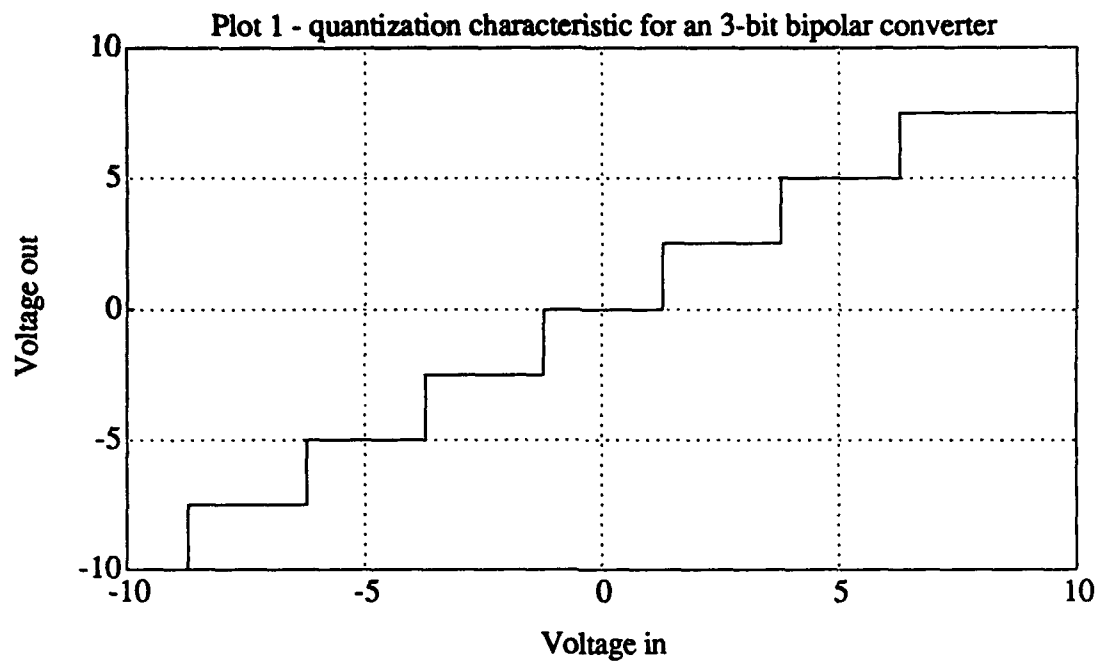
Question 4: List the amplitude ("voltage") of the 5-bit bipolar quantized signal in each of the first 5 sampling periods.

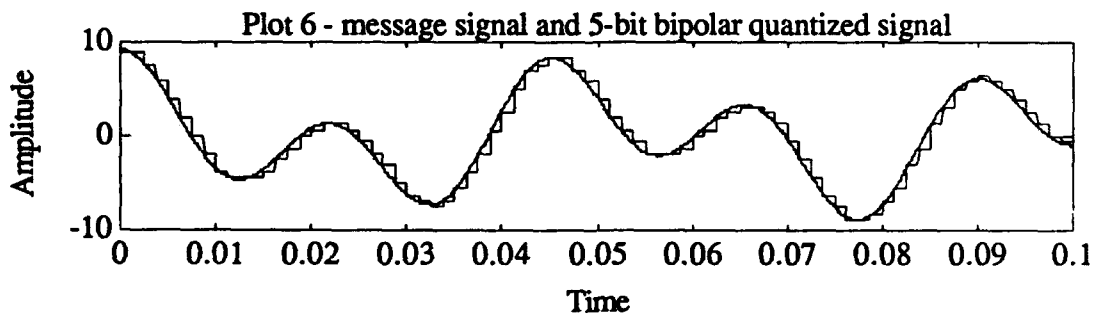
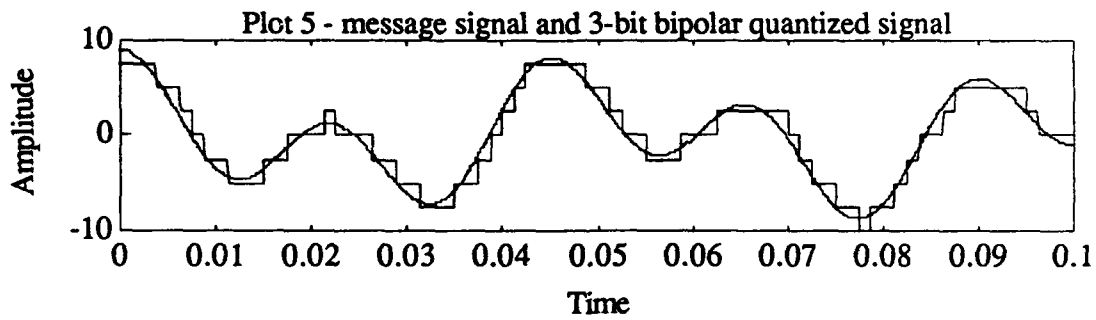
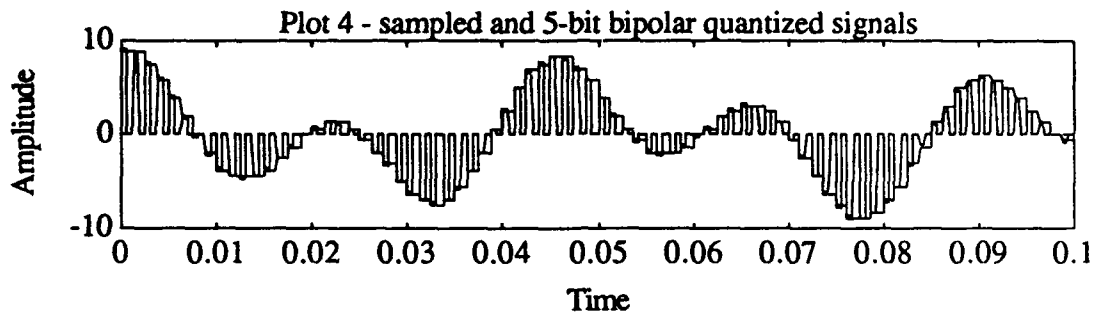
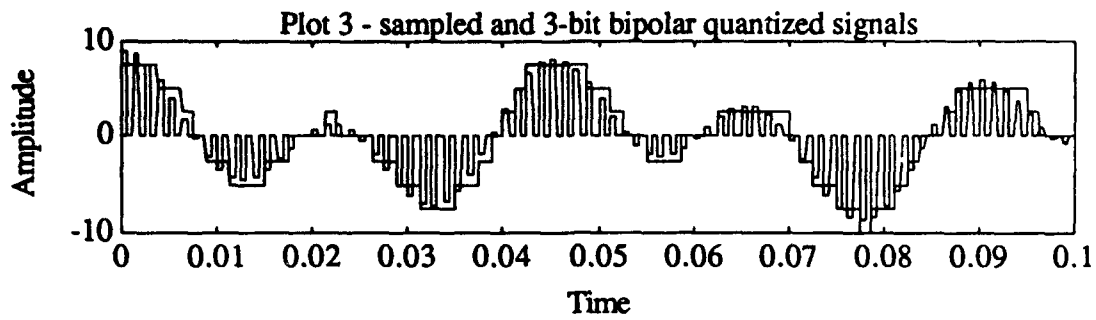
Answer: 8.75 V
8.75 V
7.5 V
5.6250 V
3.75 V

Question 5: List the values of the signal to noise ratios for the 3-bit bipolar and 5-bit bipolar quantized signals. Which converter produced less quantization noise?

Answer: 3-bit converter snr = 12.3872 dB
5-bit converter snr = 15.8277 dB

The 5-bit converter produced less noise, as evidenced by its higher signal to noise ratio, or by its closer representation to the message signal.





lab4A_ex.m

%Lab 4A example script for instructor use
%Answers will vary!

%%
%Programming Lab 4A Analog-to-Digital Conversion
% (Quantization)
%%
%Part 1--Evaluate two analog-to-digital converters
%A. Evaluate a bipolar converter

clear
clf

[quanch_x1,quanch_y1]=quantize(2,3); %generate the 3-bit converter

%Plot 1

stairs(quanch_x1,quanch_y1)
grid
title('Plot 1 - quantization characteristic for an 3-bit bipolar converter')
xlabel('Voltage in')
ylabel('Voltage out')

pause
clf

%B. Evaluate a unipolar converter

[quanch_x2,quanch_y2]=quantuni(2,5); %generate the 5-bit converter

%Plot 2

stairs(quanch_x2,quanch_y2)
grid
title('Plot 2 - quantization characteristic for 5-bit unipolar converter')
xlabel('Voltage in')
ylabel('Voltage out')

pause
clf

%%
%Part 2--Observe the quantization process
%A. Generate and sample a signal

delta_t=.0001; %set signal and sampling variables


```

d=.5;
samprate=800;

t=0:delta_t:0.1;
s=4*cos(2*pi*20*t)+5*cos(2*pi*45*t); %signal frequencies must be less than
                                     %half the sampling rate!
flatsig=flattop(s,delta_t,samprate,d);

%B. Quantize the signal using an 3-bit bipolar converter

[qx1,qy1,quan_sig1,bin_nums1]=quantize(2,3,flatsig,samprate,delta_t);

%Plot 3

subplot(211), %plot the sampled and quantized signals
plot(t,[flatsig;quan_sig1])
title('Plot 3 - sampled and 3-bit bipolar quantized signals')
xlabel('Time')
ylabel('Amplitude')

%C. Quantize the signal using a 5-bit bipolar converter

[qx2,qy2,quan_sig2,bin_nums2]=quantize(2,5,flatsig,samprate,delta_t);

%Plot 4

subplot(212),
plot(t,[flatsig;quan_sig2])
title('Plot 4 - sampled and 5-bit bipolar quantized signals')
xlabel('Time')
ylabel('Amplitude')

pause
clg

qy1
bin_nums1(1:5)

qy2
bin_nums2(1:5)

%%%%%%%%%%%%%%
%Part 3--Measure the quantization noise
%A. Find the signal to noise ratios

%Plot 5

subplot(211),

```

```
plot(t,[s;quan_sig1])  
title('Plot 5 - message signal and 3-bit bipolar quantized signal')  
xlabel('Time')  
ylabel('Amplitude')
```

%Plot 6

```
subplot(212),  
plot(t,[s;quan_sig2])  
title('Plot 6 - message signal and 5-bit bipolar quantized signal')  
xlabel('Time')  
ylabel('Amplitude')
```

%B. Compare quantization noise for the two systems

```
snr_3bit=snr(s,quan_sig1)  
snr_5bit=snr(s,quan_sig2)
```

**This page is intentionally
left blank.**

EO 3513 Programming Laboratory 4B Key

Pulse Code Modulation (PCM)

Answers will vary. The answers below are based on the following signal:

$$s=7*\cos(2*\pi*10*t)+3*\cos(2*\pi*35*t);$$

Question 1: In your own words, describe the NRZL unipolar encoding scheme.

Answer: Marks are indicated by positive voltage; spaces are indicated by zero voltage.

Question 2: Predict an adequate baseband signal bandwidth for the NRZL unipolar coded signal based on its spectral plot.

An. Prediction: B = approximately 1000 Hz

Question 3: In your own words, describe the RZL unipolar encoding scheme.

Answer: Marks are indicated by positive voltage; spaces are indicated by zero voltage. Voltage always drops to zero for the last half of the bit duration.

Question 4: Predict an adequate baseband signal bandwidth for the RZL unipolar coded signal based on its spectral plot.

Answer: Prediction: B = approximately 2000 Hz

Question 5: In your own words, describe the manchester encoding scheme.

Answer: Changes are indicated by a transition in the middle of the bit. Marks always change from high to low voltage; spaces change from low to high voltage.

Question 6: Predict an adequate baseband signal bandwidth for the NRZL unipolar coded signal based on its spectral plot.

Answer: Prediction: B = approximately 2500 Hz

Question 7: Record the values of "codedsig." Is this bit pattern reflected on Plots 2, 4, and 6?

Answer:

1	1	1	1	1	1	1	1	1	0	1	0
1	0	1	1	0	0	1	0	0	1	0	0
1	0	0	1	0	0	1	0	0	1	1	1
1	0	0	0	1	0	0	1	0	1	0	0
0	0	0	1	1	0	0	0	0	0	1	0

Plots reflect the bit pattern above.

Question 8: Calculate the approximate baseband bandwidth for the PCM signals:

NRZL unipolar coded signal
RZL and manchester coded signals

How do these values compare with your predictions?

Answer: For NRZL signal: $B = 0.5/\tau \Rightarrow 1/0.0083 \text{ seconds} \Rightarrow 600 \text{ Hz}$

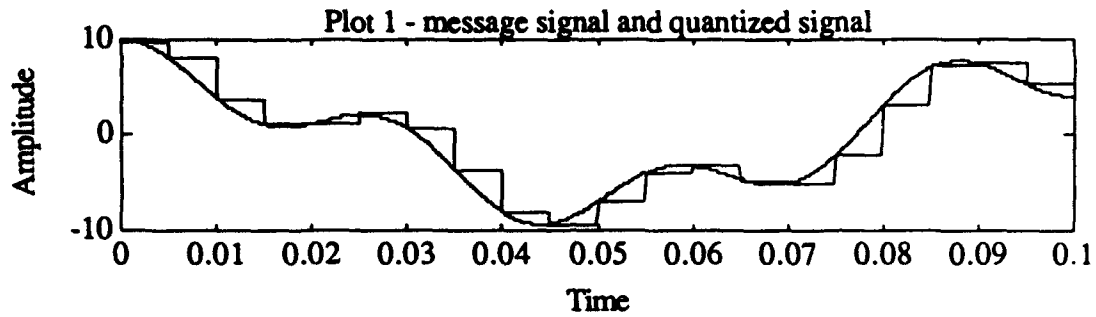
For RZL and manchester signals: $B = 0.5/\tau \Rightarrow 1/0.00042 \text{ seconds} \Rightarrow 1200 \text{ Hz}$

The predicted NRZL signal bandwidth of 1000 Hz is within 400 Hz of the calculation.

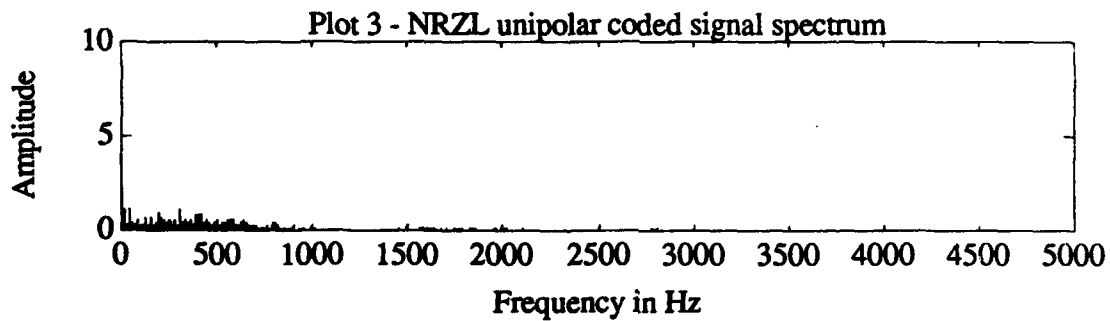
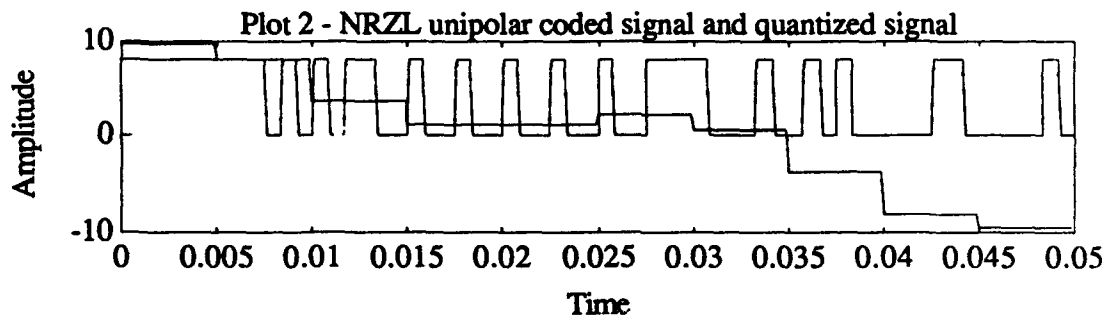
The predicted RZL signal bandwidth of 2000 Hz is within 800 Hz of the calculation.

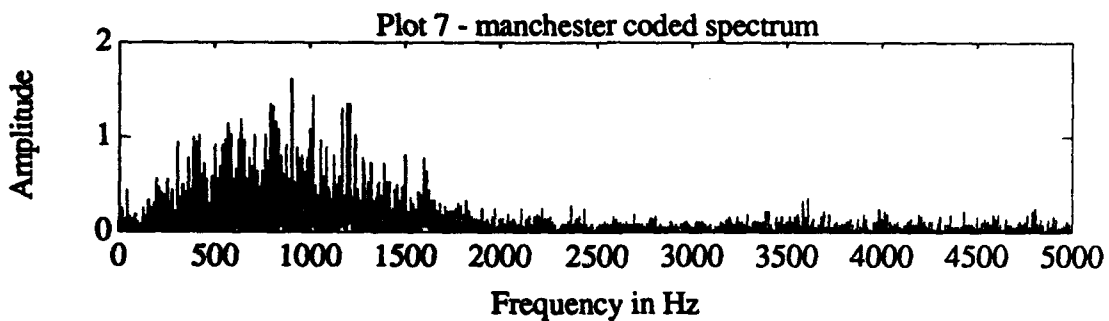
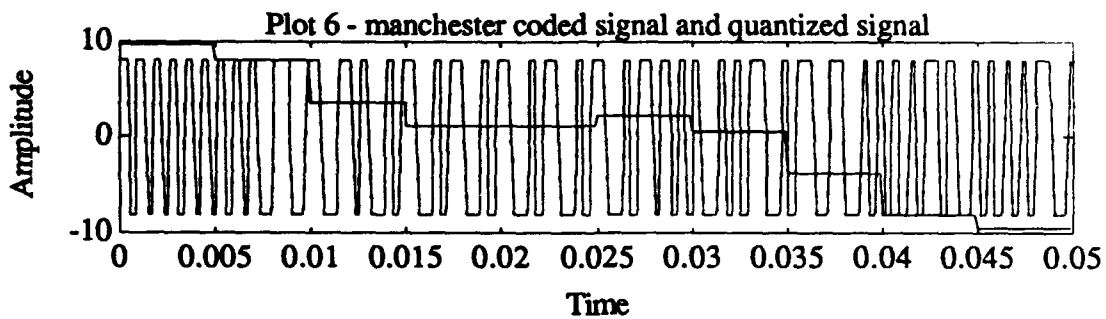
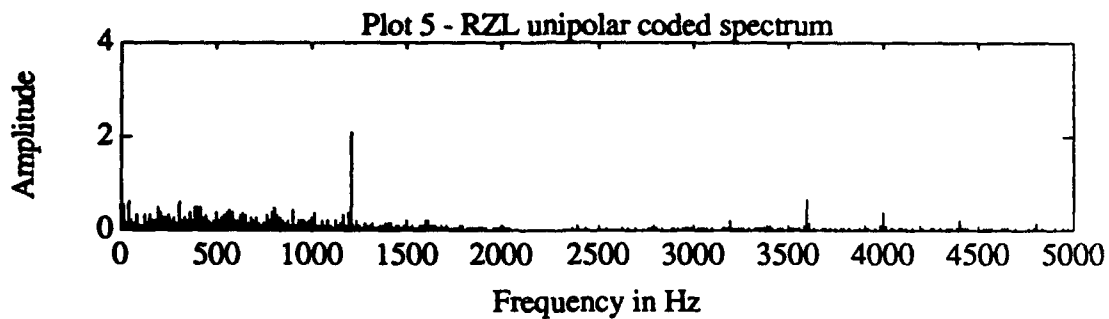
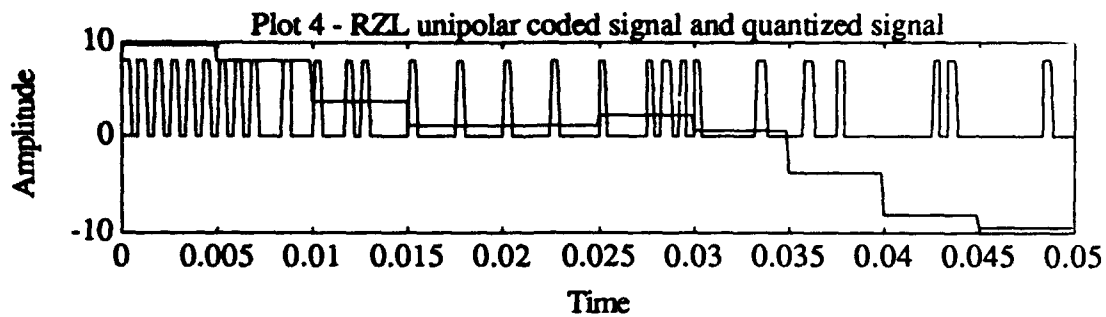
The predicted manchester signal bandwidth of 2500 Hz is more than twice the calculation.

Predictions are conservative in all cases.



NO PLOT HERE--JUST PRESS RETURN





lab4B_ex.m

%Lab 4B example script for instructor use
%Answers will vary!

%%
%Programming Lab 4B Digital Encoding
%%
%Part 1--Generate a bitstream to encode
%A. Generate a message signal

clear
clg

delta_t=.0001; %set signal and sampling variables
samprate=200;

t=0:delta_t:1;
s=7*cos(2*pi*10*t)+3*cos(2*pi*35*t);

%Plot 1

subplot(211),
plot(t(1:1000),s(1:1000))
title('Plot 1 - message signal and quantized signal')
xlabel('Time')
ylabel('Amplitude')
hold on

%B. Quantize the message signal

elements=6;

[qx,qy,quan_sig,bin_nums]=quantize(2,elements,s,samprate,delta_t);

plot(t,quan_sig,'b')
hold off

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

pause
clg

clear s;clear qx;clear qy;

%C. Generate a bitstream


```

codedsig=encode(bin_nums,2,elements); %binary-encode the signal

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Part 2--Encode the bitstream using pulse code modulation (PCM)
%A. Generate a non-return-to-zero level (NRZL) unipolar
% coded signal and spectrum

bitrate=samprate*elements;

nrzlunisig=nrzluni(codedsig,delta_t,bitrate);
nrzlunisig=8*nrzlunisig;

%Plot 2

subplot(211),
plot(t(1:500),[quan_sig(1:500);nrzlunisig(1:500)])
title('Plot 2 - NRZL unipolar coded signal and quantized signal')
xlabel('Time')
ylabel('Amplitude')

[specman,HZ]=spectral(nrzlunisig,delta_t);

%Plot 3

subplot(212),
plot(HZ,specman)
title('Plot 3 - NRZL unipolar coded signal spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

clear specman
clear nrzlunisig

%C. Generating a return-to-zero level (RZL) unipolar coded signal

rzunisig=rzuni(codedsig,delta_t,bitrate);
rzunisig=8*rzunisig;

%Plot 4

subplot(211),
plot(t(1:500),[quan_sig(1:500);rzunisig(1:500)])
title('Plot 4 - RZL unipolar coded signal and quantized signal')
xlabel('Time')
ylabel('Amplitude')

```

```

specman=spectral(rzunisig,delta_t);

%Plot 5

subplot(212),
plot(Hz,specman)
title('Plot 5 - RZL unipolar coded spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

clear specman
clear rzunisig

%D. Generating a manchester coded signal

manchsig=manchest(codedsig,delta_t,bitrate);
manchsig=8*manchsig;

%Plot 6

subplot(211),
plot(t(1:500),[quan_sig(1:500);manchsig(1:500)])
title('Plot 6 - manchester coded signal and quantized signal')
xlabel('Time')
ylabel('Amplitude')

specman=spectral(manchsig,delta_t);

%Plot 7

subplot(212),
plot(Hz,specman)
title('Plot 7 - manchester coded spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

codedsig(1:60)

```

**This page is intentionally
left blank.**

EO 3513 Programming Laboratory 4C Key *Companding*

Answers will vary according to student's choices for values of sampling rate and mu.

Question 1: What is the effect on the compressed signal of increasing the value of mu?

Answer: Increasing the value of mu increases the compression function, raising the lower signal values more, which minimizes the extreme differences in the signal.

Question 2: Obtain the value of the signal to noise ratio (no companding). Record this value. By exceeding this benchmark value, you will be decreasing quantization noise.

Answer: snr_quant = 8.0639 dB

Question 3: Obtain the value of the signal to noise ratio using companding. Record this value.

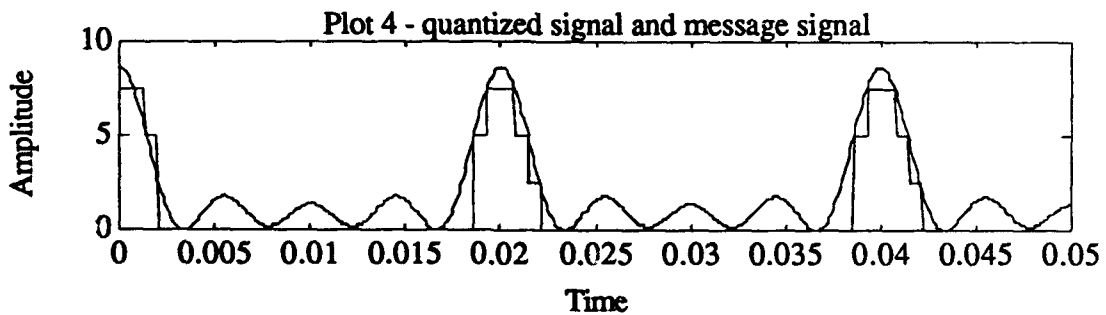
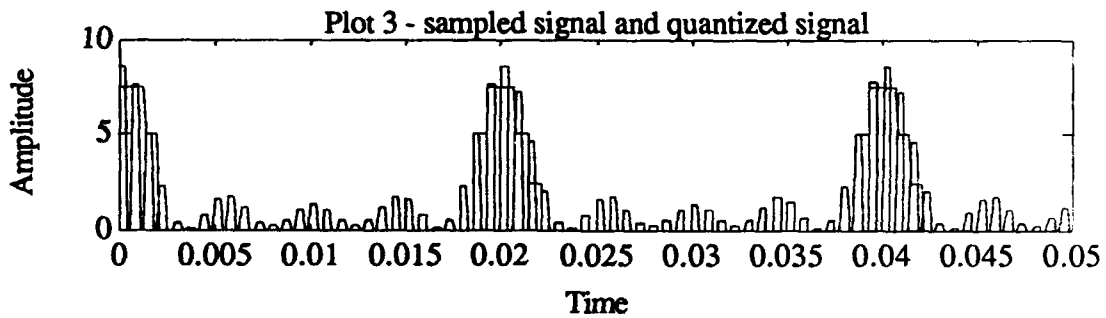
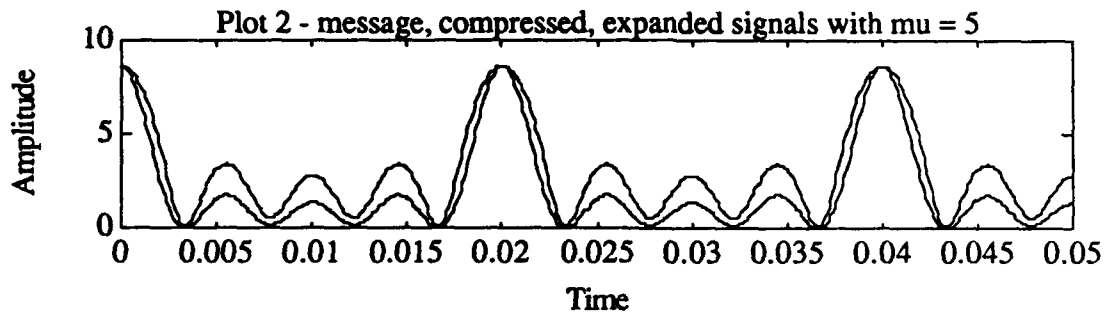
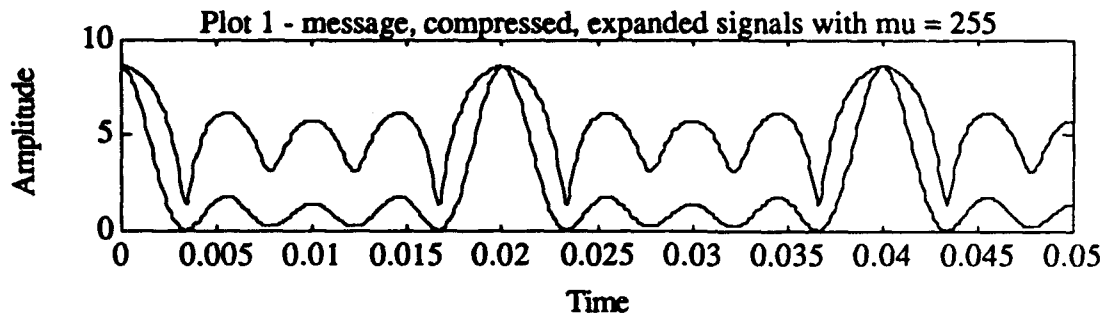
Answer: snr_cmpnd = 9.18 dB

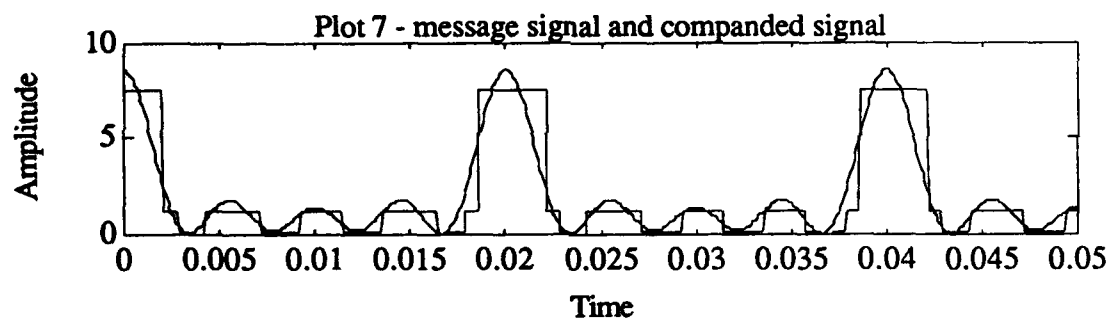
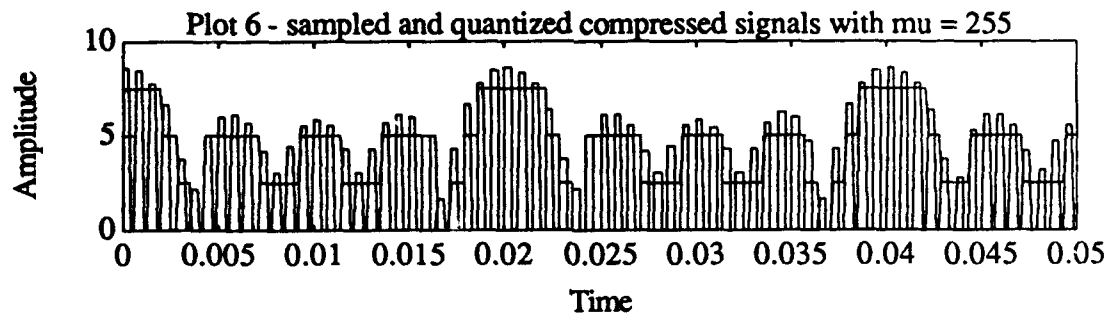
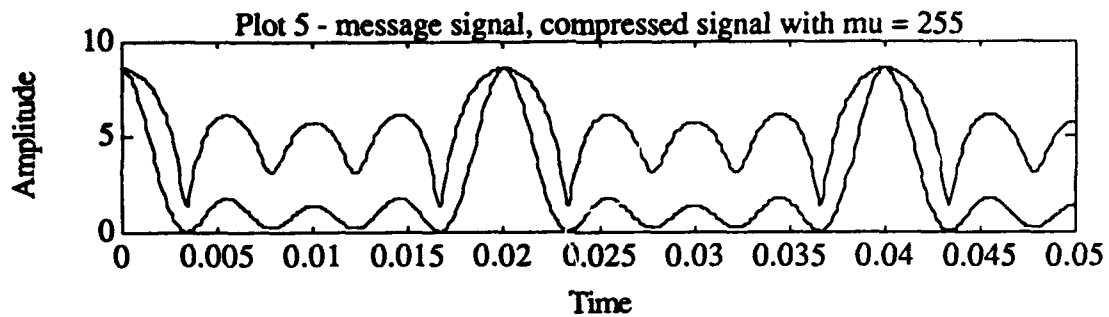
Question 4: Provide the following characteristics of your system:

sampling rate
value of mu

Answer: sampling rate = 1400 Hz
mu = 255

Note: In general, lower sampling rates require lower values of mu in order to reduce quantization noise through companding the message signal. For example, with a sampling rate of 500 Hz, mu must be equal to or less than 50. With a value of mu = 255, the sampling rate must be at least 780 Hz.





NO PLOT HERE

lab4C_ex.m

```
%Lab 4C example script for instructor use
%Answers will vary!

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Programming Lab 4C--Companding
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Part 1--Observe the effects of companding on the quantization
%   process
%A. Generate the signal

clear
clg

delta_t=.0001;
t=0:delta_t:0.05; %no spectral analysis, so don't need a long
                  %vector

s=2+2.1*cos(2*pi*50*t)+1.7*cos(4*pi*50*t)+1.5*cos(6*pi*50*t)...
+1.3*cos(8*pi*50*t);

%B. Compare compression characteristics for high and low values of mu

muhigh=255; %highest value of mu
mulow=5; %lowest value of mu

sighigh=compress(s,muhigh,max(s)); %compress with mu=255
exhigh=expand(sighigh,muhigh,max(sighigh)); %expand with mu=255

%Plot 1

subplot(211),
plot(t,[s;sighigh])
title('Plot 1 - message, compressed, expanded signals with mu = 255')
xlabel('Time')
ylabel('Amplitude')
hold on
pause(3)
plot(t,exhigh,'b')
hold off

siglow=compress(s,mulow,max(s)); %compress with mu=5
exlow=expand(siglow,mulow,max(siglow)); %expand with mu=5

%Plot 2

subplot(212),
```

```

plot(t,[s;siglow])
title('Plot 2 - message, compressed, expanded signals with mu = 5')
xlabel('Time')
ylabel('Amplitude')
hold on
pause(3)
plot(t,exlow,'b')
hold off

pause
clg

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Part 2--Reduce quantization noise by companding the message signal
%A. Sample and quantize the message signal

samprate=1400; %set signal and sampling variables
d=.5;
mu=255;

flatsig=flatop(s,delta_t,samprate,d);
[qx,qy,quansig]=quantuni(2,2,flatsig,samprate,delta_t);

%Plot 3

subplot(211),
plot(t,[flatsig;quansig])
title('Plot 3 - sampled signal and quantized signal')
xlabel('Time')
ylabel('Amplitude')

%Plot 4

subplot(212),
plot(t,[quansig;s])
title('Plot 4 - quantized signal and message signal')
xlabel('Time')
ylabel('Amplitude')

pause
clg

%B. Establish a benchmark for signal to noise ratio

snr_quant=snr(s,quansig) %find the quantization noise

%C. Compress, sample, and quantize the message signal

```



```

compsig=compress(s,mu,max(s)); %compress signal

%Plot 5

subplot(211),
plot(t,[s;compsig])
title('Plot 5 - message signal, compressed signal with mu = 255')
xlabel('Time')
ylabel('Amplitude')

flatcomp=flattop(compsig,delta_t,samprate,d); %sample the compressed signal
[qx,qy,quancomp]=quantuni(2,2,flatcomp,samprate,delta_t);

%Plot 6

subplot(212),
plot(t,[flatcomp;quancomp])
title('Plot 6 - sampled and quantized compressed signals with mu = 255')
xlabel('Time')
ylabel('Amplitude')

pause
clf

%D. Expand the quantized compressed signal

cmpndsig=expand(quancomp,mu,max(quancomp));

%Plot 7

subplot(2;1),
plot(t,[s;cmpndsig])
title('Plot 7 - message signal and companded signal')
xlabel('Time')
ylabel('Amplitude')

%E. Find the signal to noise ratio for the companded signal

snr_cmpnd=snr(s,cmpndsig)

subplot(212),
title('NO PLOT HERE')

```

EO 3513 Programming Laboratory 5 Key *Amplitude Modulation Double Sideband (AM DSB)*

Answers will vary. The answers below are based on the following signals:

$$s_{gl}=5\cos(2\pi\cdot 200\cdot t);$$

$$m_{lt}=5\cos(2\pi\cdot 400\cdot t)+3\cos(2\pi\cdot 100\cdot t)+2\cos(2\pi\cdot 350\cdot t);$$

Question 1: Calculate the following values for the single-tone message signal:

**peak power
average power
baseband bandwidth**

Answer: peak power = $P_P = \frac{A_P^2}{2} \Rightarrow 5^2 / 2 \Rightarrow 12.5$

average power = $P = A_0^2 + \frac{1}{2} \sum_{N=1}^{\infty} (A_N^2 + B_N^2) \Rightarrow 5^2 / 2 \Rightarrow 12.5$

baseband bandwidth = 200 Hz

Question 2: Predict the following values for the single-tone AM DSB signal:

**peak power
average power
transmission bandwidth**

Answer: peak power = $P_P = \frac{A_P^2}{2} \Rightarrow 5^2 / 2 \Rightarrow 12.5$

average power = $P = \frac{A^2}{4} \Rightarrow 5^2 / 4 \Rightarrow 6.25$

transmission bandwidth = 400 Hz

Question 3: Record the values representing peak and average power for the signal-tone signal.

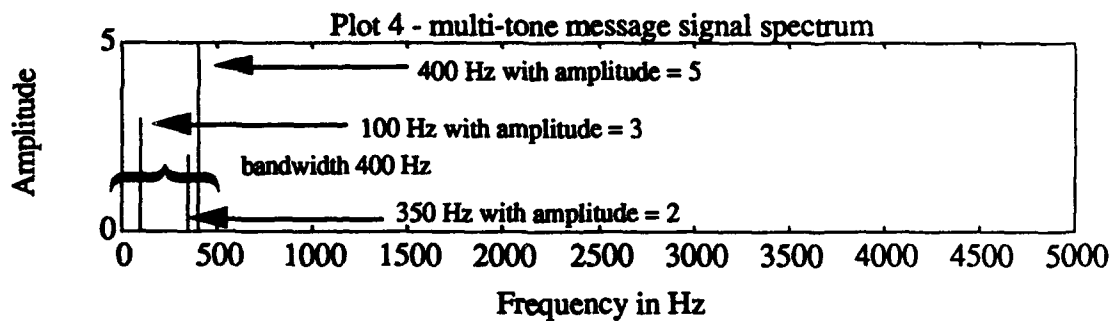
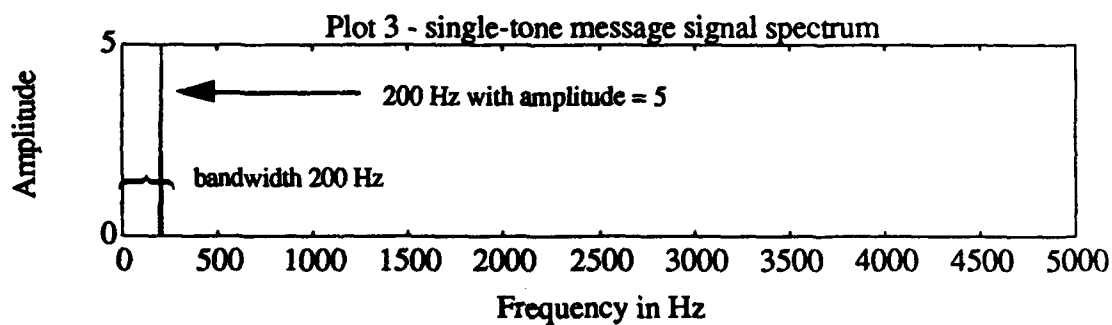
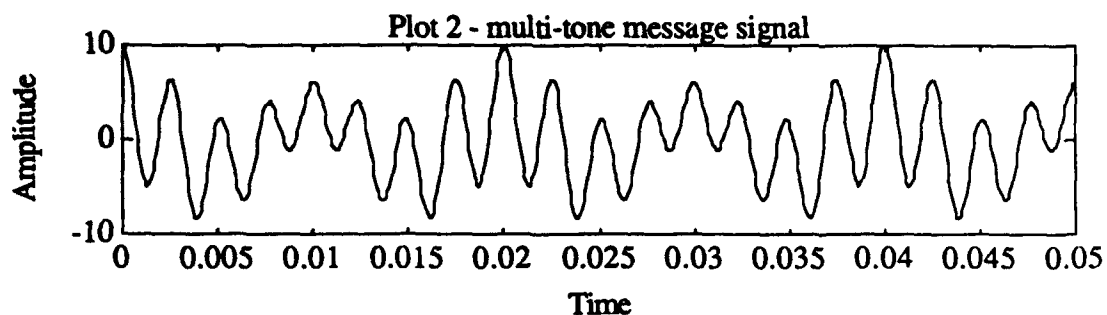
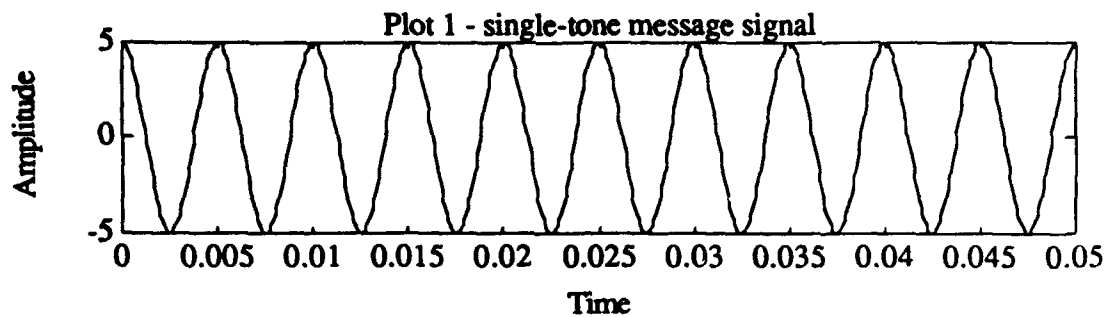
Do your calculations for bandwidth and power in Question 3 agree with the computer-generated values?

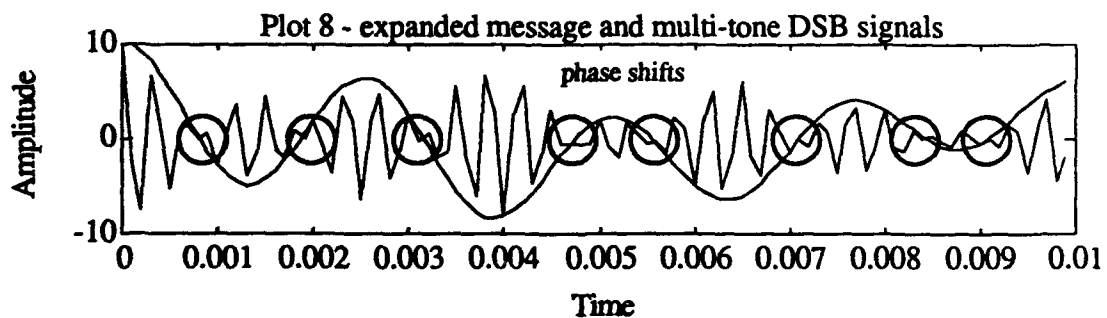
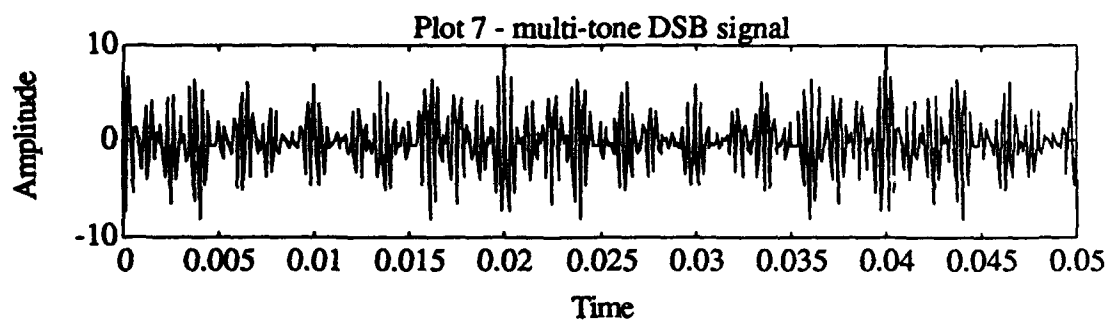
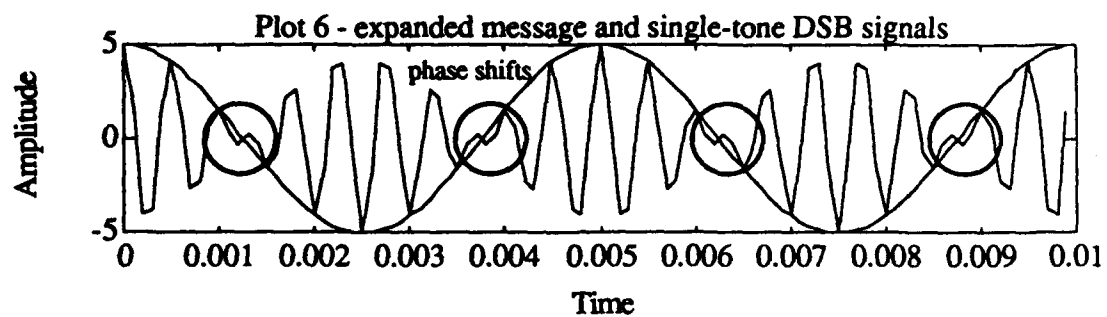
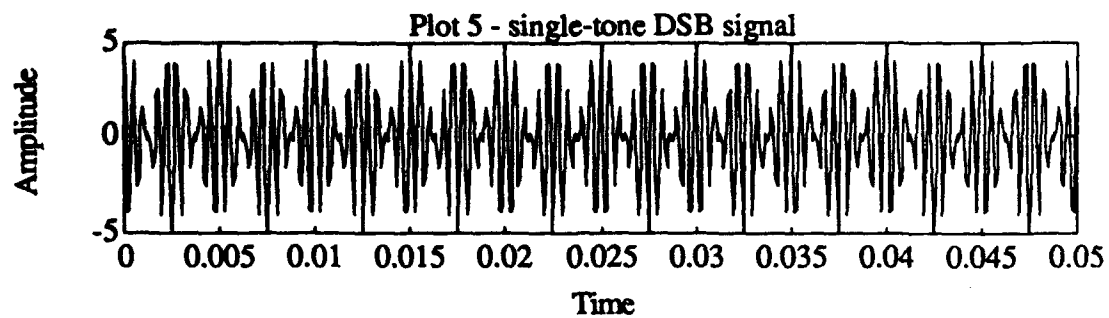
Answer: $\text{dsb_pk_pwr_sngl} = 12.5$
 $\text{dsb_avg_pwr_sngl} = 6.2506$

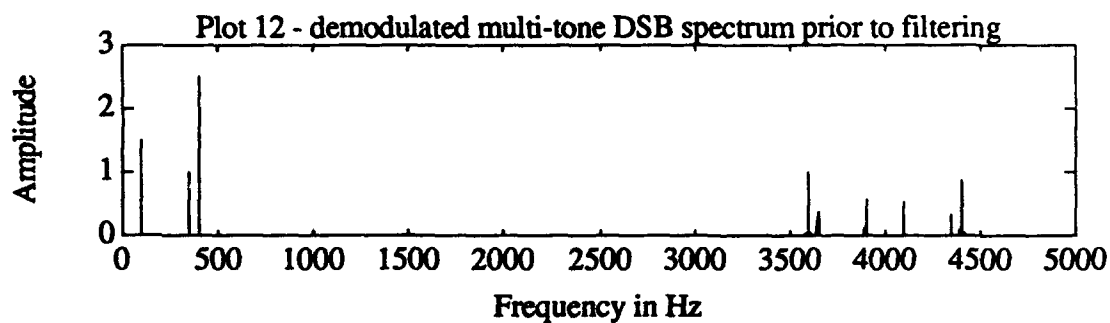
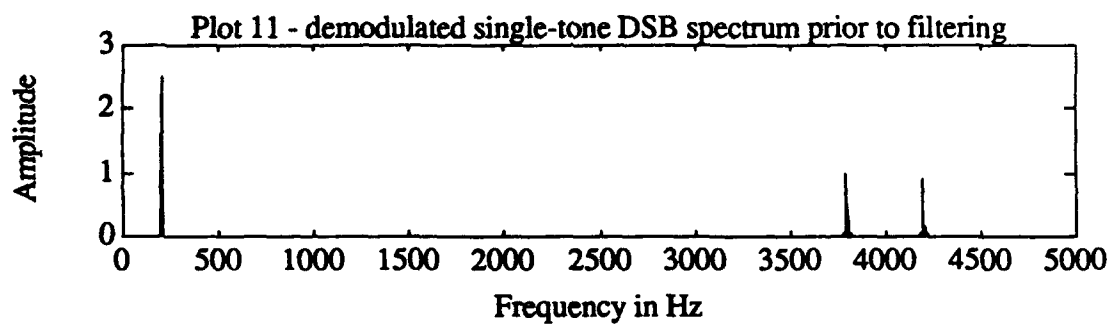
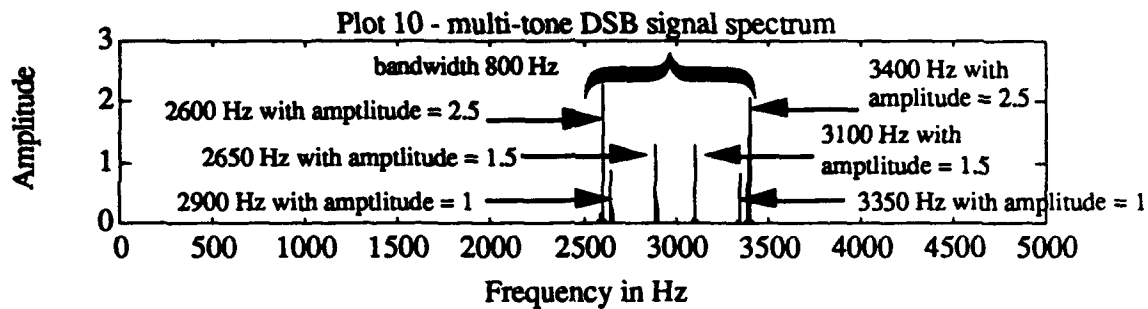
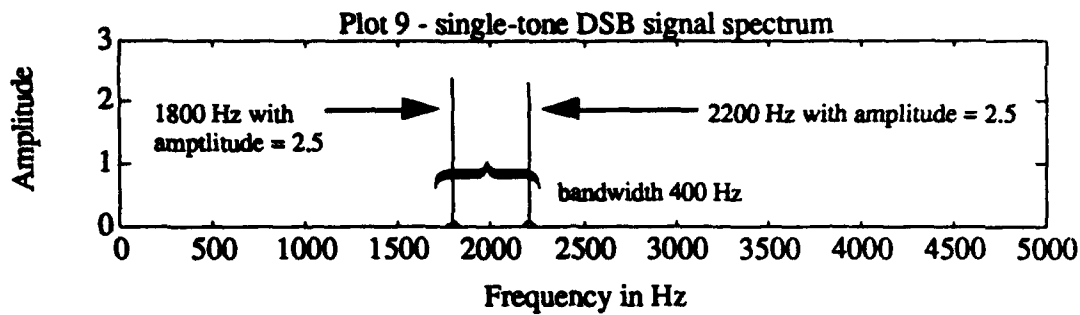
Yes—calculations agree.

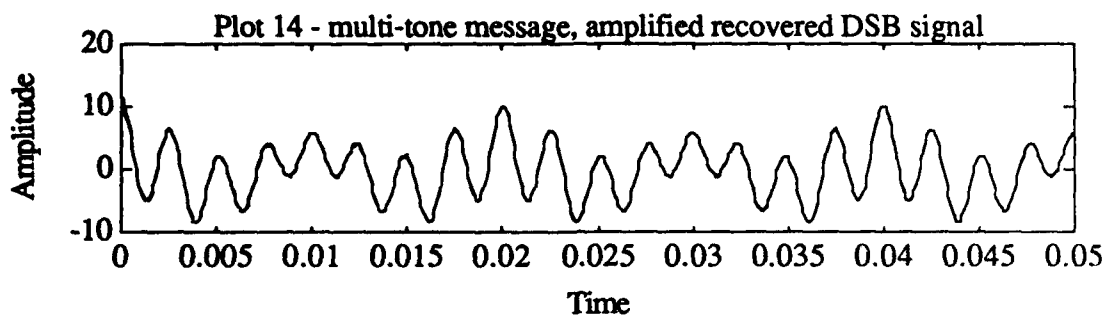
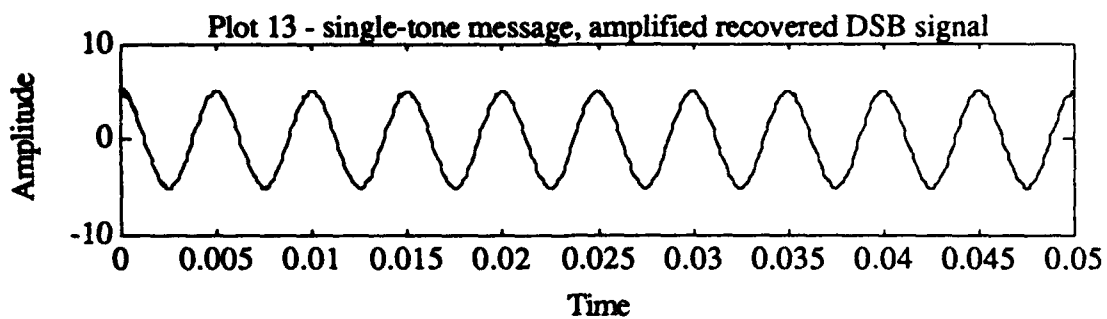
Question 4: Why is coherent detection (detection using the carrier) necessary for an AM DSB signal?

Answer: Envelope detection of the AM DSB signal would not detect phase shifts, which indicate that the message signal has changed from positive to negative values, or from negative to positive values.









lab5_ex.m

%Programming Lab 5 example for instructor use

%Answers will vary!

%%

%Programming Lab 5 Amplitude Modulation AM DSB

%%

%Part 1--Generate single- and multi-tone message signals

%and spectra

%A. Generating the message signals

clear

clg

delta_t=.0001; %set signal and sampling variables

t=0:delta_t:1;

sgl=5*cos(2*pi*200*t); %single-tone signal variable

%multi-tone signal variable

mlt=5*cos(2*pi*400*t)+3*cos(2*pi*100*t)+2*cos(2*pi*350*t);

%Plot 1

subplot(211), %plot the single-tone signal

plot(t(1:500),sgl(1:500))

title('Plot 1 - single-tone message signal')

xlabel('Time')

ylabel('Amplitude')

%Plot 2

subplot(212), %plot the multi-tone signal

plot(t(1:500),mlt(1:500))

title('Plot 2 - multi-tone message signal')

xlabel('Time')

ylabel('Amplitude')

pause

clg

%B. Predict power and bandwidth for the message signals

%C. Verify bandwidth for the message signals

[specsgl,HZ]=spectral(sgl,delta_t); %generate the spectrum

Programming Laboratory 5 Key—page 7


```

%Plot 3

subplot(211), %plot the spectrum
plot(Hz,specsgl)
title('Plot 3 - single-tone message signal spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

[specmlt,HZ]=spectral(mlt,delta_t); %generate the spectrum

%Plot 4

subplot(212), %plot the spectrum
plot(Hz,specmlt)
title('Plot 4 - multi-tone message signal spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear specmlt; clear specsgl;

pause
clg

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Part 2--Generate AM DSB signals using single- and multi-tone
%input

%A. Generate the single-tone AM DSB signal and spectrum

moddsbsgl=cos(2*pi*2000*t).*sgl; %modulate the signal by multiplying by a cosine
moddsbmlt=cos(2*pi*3000*t).*mlt;

%Plot 5

subplot(211), %plot the modulated signal
plot(t(1:500),moddsbsgl(1:500))
title('Plot 5 - single-tone DSB signal')
xlabel('Time')
ylabel('Amplitude')

%Plot 6

subplot(212), %plot detailed view to show phase shifts
plot(t(1:100),[sgl(1:100);moddsbsgl(1:100)])
title('Plot 6 - expanded message and single-tone DSB signals')
xlabel('Time')
ylabel('Amplitude')

```

```
pause
clg
```

```
%Plot 7
```

```
subplot(211), %plot the modulated signal
plot(t(1:500),moddsbmlt(1:500))
title('Plot 7 - multi-tone DSB signal')
xlabel('Time')
ylabel('Amplitude')
```

```
%Plot 8
```

```
subplot(212), %plot detailed view to show phase shifts
plot(t(1:100),[mlt(1:100);moddsbmlt(1:100)])
title('Plot 8 - expanded message and multi-tone DSB signals')
xlabel('Time')
ylabel('Amplitude')
```

```
pause
clg
```

```
%B. Predict power for the single-tone AM DSB signal
```

```
%C. Verify bandwidth for the single- and multi-tone AM DSB signals
```

```
dsbsglspec=spectral(moddsbsgl,delta_t); %generate the modulated spectrum
dsbmllspec=spectral(moddsbmlt,delta_t);
```

```
%Plot 9
```

```
subplot(211), %plot the modulated spectrum
plot(Hz,dsbsglspec)
title('Plot 9 - single-tone DSB signal spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')
```

```
%Plot 10
```

```
subplot(212), %plot the modulated spectrum
plot(Hz,dsbmllspec)
title('Plot 10 - multi-tone DSB signal spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')
```

```
clear dsbmllspec;clear dsbsglspec;
```

```

pause
clg

%D. Verify the power for the single-tone AM DSB signal

dsb_pk_pwr_sngl=((max(moddsbsgl))^2)/2 %find the peak power

psddsb=psd(moddsbsgl,delta_t); %generate the power spectral density

dsb_avg_pwr_sngl=sum(psddsb) %find average power by summing the power
                        %spectral density values

clear psddsb;

%%%%%%%%%%%%%%
%Part 3--Recover the AM DSB signals
%A. Demodulate the AM DSB signals

demodsgl=cos(2*pi*2000*t).*moddsbsgl; %first step in recovering the signal--
                        %multiply by the carrier
demodmlt=cos(2*pi*3000*t).*moddsbmlt;

[specsgl,HZ,sglfft]=spectral(demodsgl,delta_t); %generate the spectrum
[specmlt,HZ,mltfft]=spectral(demodmlt,delta_t); %the recovered signal

clear demodsgl;clear demodmlt;clear moddsbsgl;clear moddsbmlt;

%Plot 11

subplot(211),
plot(HZ,specsgl)
title('Plot 11 - demodulated single-tone DSB spectrum prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')

%Plot 12

subplot(212),
plot(HZ,specmlt)
title('Plot 12 - demodulated multi-tone DSB spectrum prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear specmlt; clear specsgl;

pause
clg

%B. Filter and recover the message signals

```

```

recsgl=recoverm(sglfft,'ideallow',Hz,250); %recover and filter
clear sglfft;

recmlt=recoverm(mltfft,'ideallow',Hz,450);
clear mltfft;

bigsgl=recsgl*2; %amplify signal
bigmlt=recmlt*2;

%Plot 13

subplot(211), %plot the recovered signal on top of the message signal
plot(t(1:500),[sgl(1:500);bigsgl(1:500)])
title('Plot 13 - single-tone message, amplified recovered DSB signal')
xlabel('Time')
ylabel('Amplitude')

%Plot 14

subplot(212), %plot the recovered signal on top of the message signal
plot(t(1:500),[mlt(1:500);bigmlt(1:500)])
title('Plot 14 - multi-tone message, amplified recovered DSB signal')
xlabel('Time')
ylabel('Amplitude')

```

**This page is intentionally
left blank.**

EO 3513 Programming Laboratory 6 Key *Amplitude Modulation Single Sideband (AM SSB)*

Answers will vary. The answers below are based on the following signals:

$$\text{sgl}=8*\cos(2*\pi*220*t);$$

$$\text{mlt}=2*\cos(2*\pi*150*t)+4*\cos(2*\pi*225*t)+8*\cos(2*\pi*400*t);$$

Question 1: Calculate the following values for the single-tone message signal:

**peak power
average power
baseband bandwidth**

Answer: $\text{peak power} = A_p^2 / 2 \Rightarrow 8^2 / 2 = 32$
 $\text{average power} = A_0^2 + 1/2 \sum (A_n^2 + B_n^2) \Rightarrow 8^2 / 2 = 32$
 $\text{baseband bandwidth} = 220 \text{ Hz}$

Question 2: Predict the following values for the single-tone AM SSB signal:

**peak power
average power
transmission bandwidth**

Answer: $\text{peak power} = A_c^2 / 2 \Rightarrow 4^2 / 2 = 8$
 $\text{average power} = A^2 / 2 \Rightarrow 4^2 / 2 = 8$
 $\text{transmission bandwidth} = 440 \text{ Hz}$

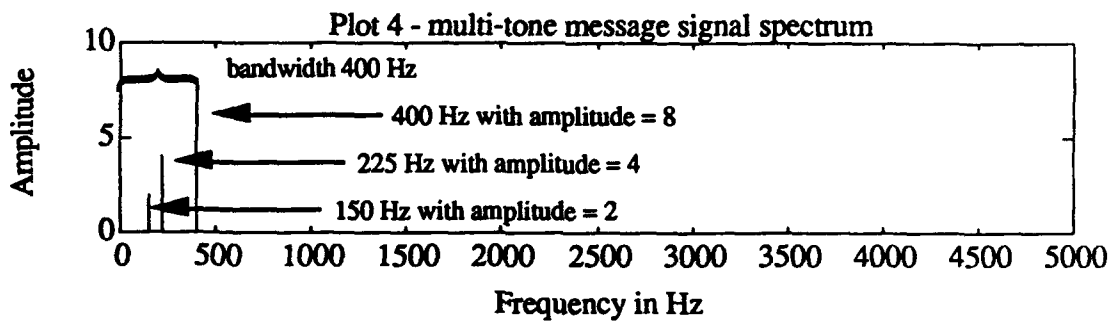
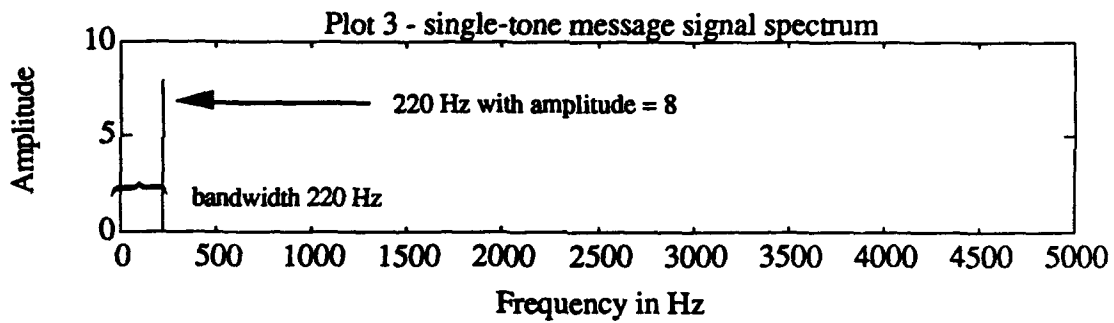
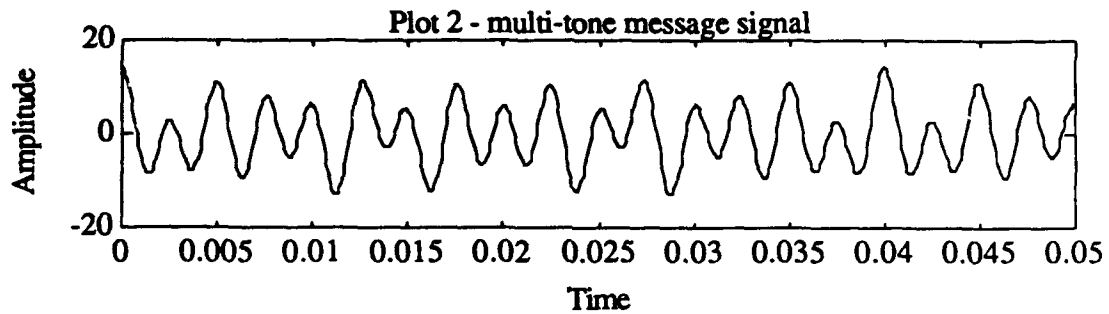
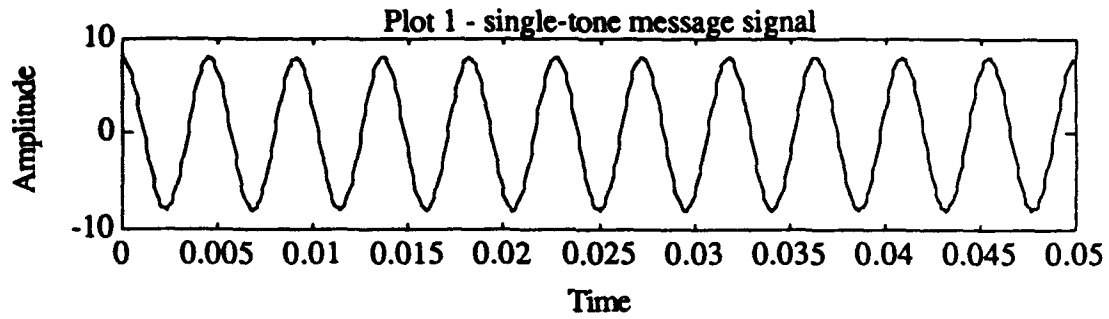
Question 3: Record the values representing peak and average power for the signal-tone LSB and USB signals.

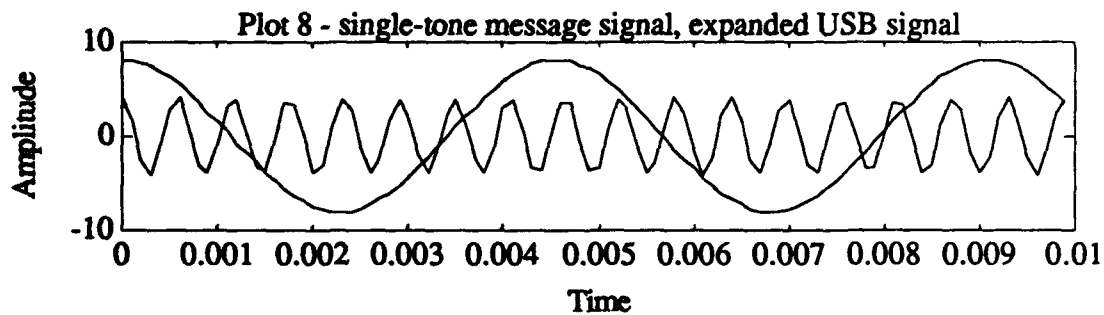
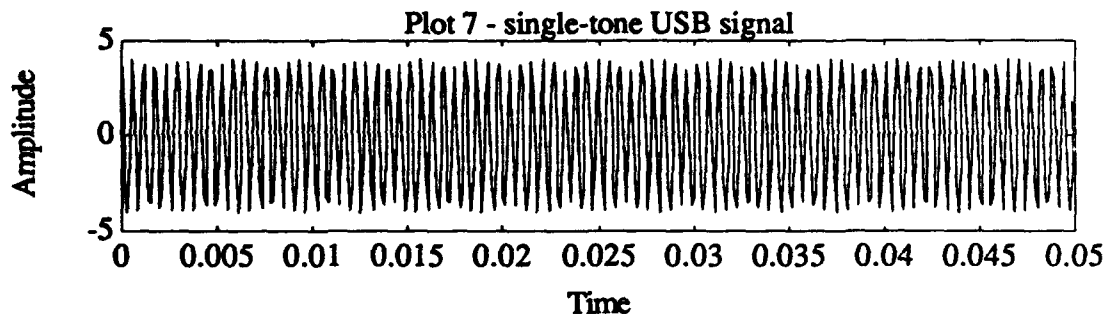
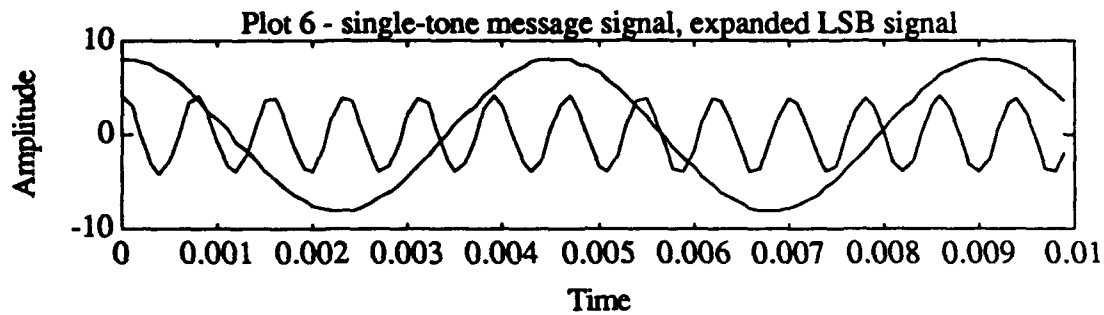
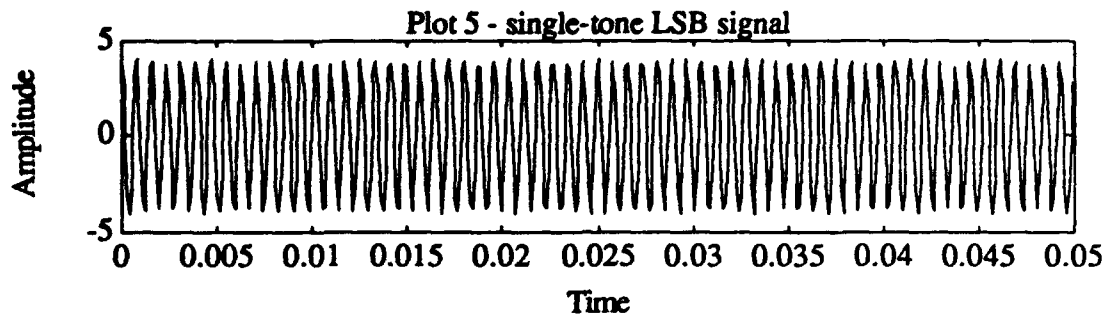
Do your calculations for bandwidth and power in Question 2 agree with the computer-generated values?

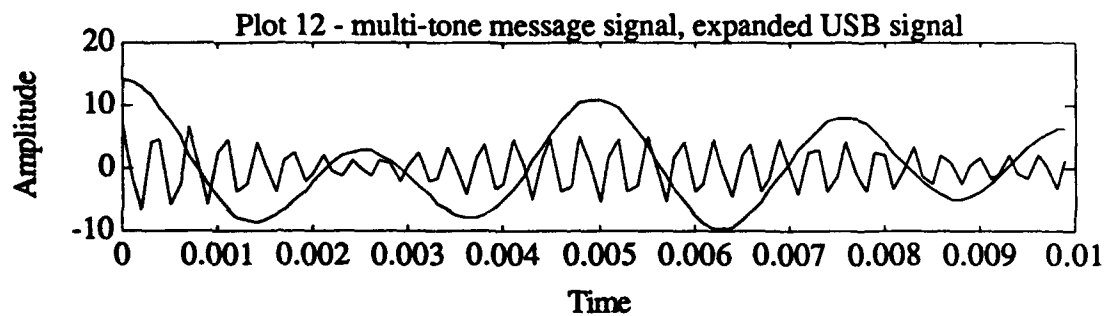
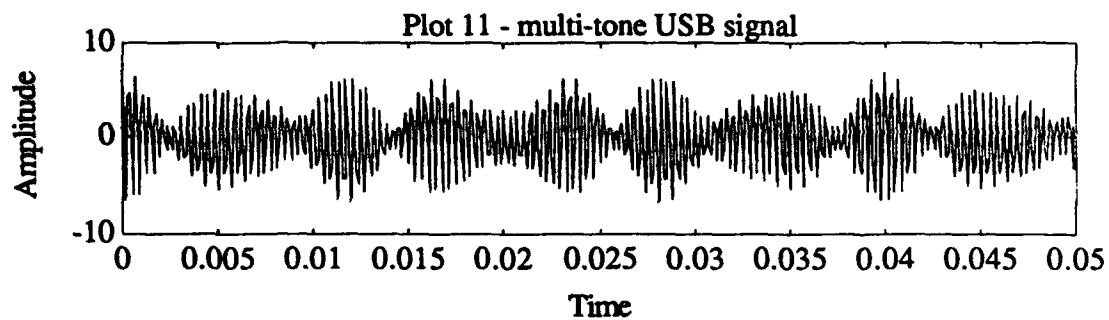
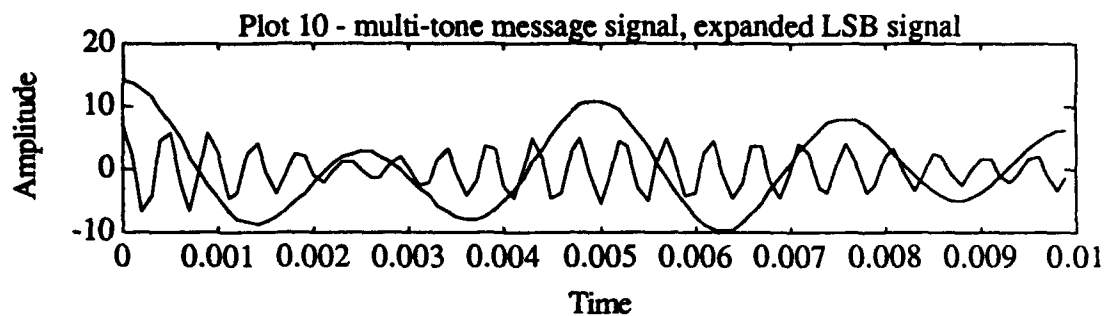
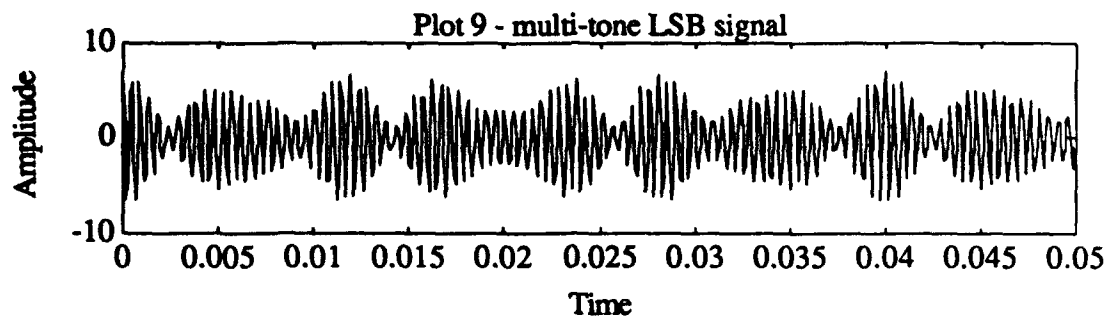
Answer: $\text{lsb_pk_pwr_sngl} = 8.2009$
 $\text{usb_pk_pwr_sngl} = 8.0261$
 $\text{lsb_avg_pwr_sngl} = 8.0001$
 $\text{usb_avg_pwr_sngl} = 7.9999$

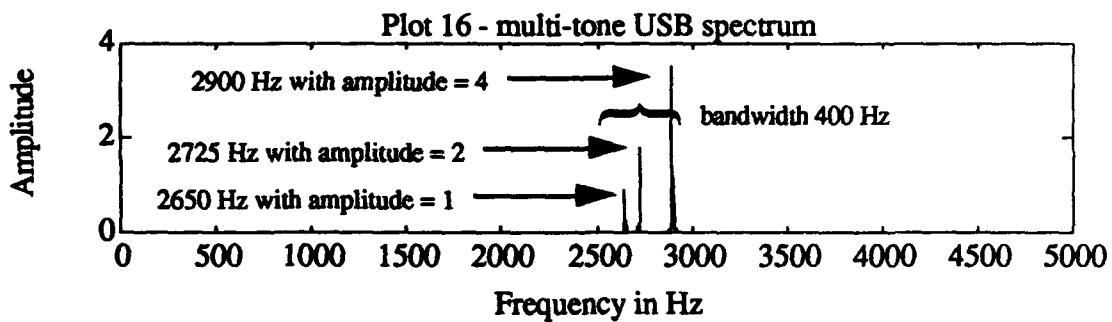
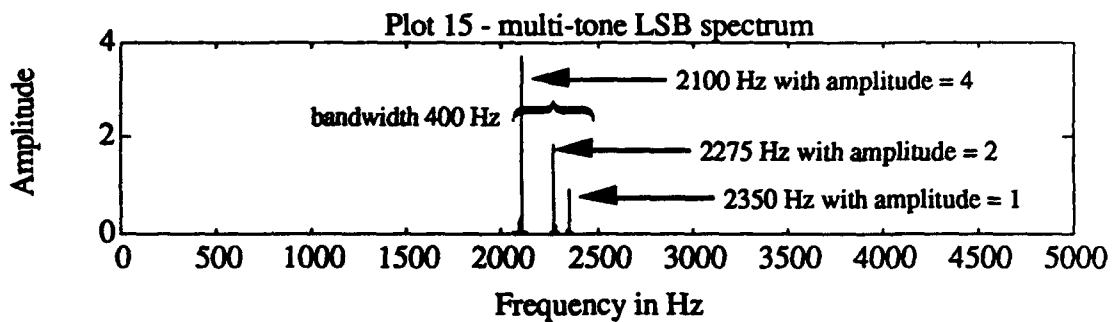
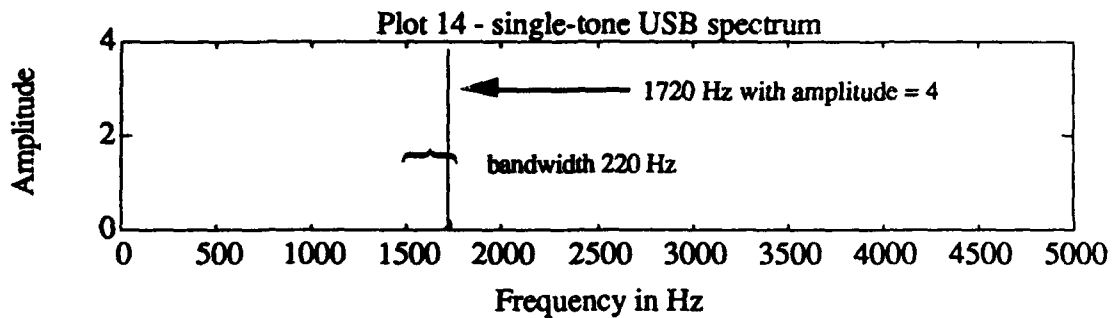
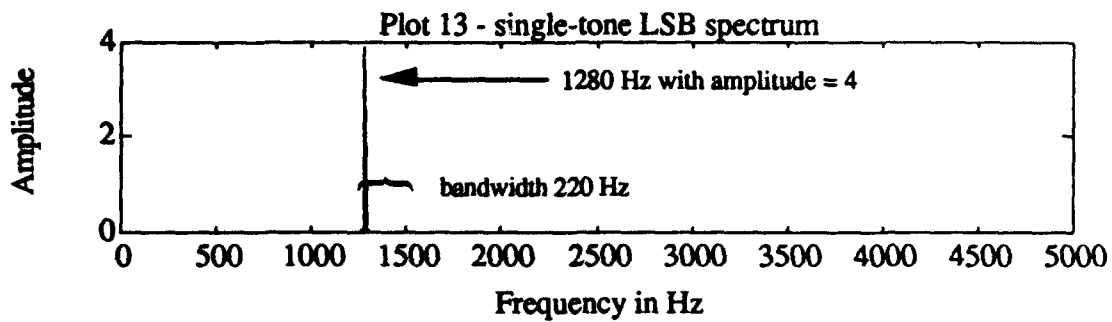
Question 4: Is coherent detection (detection using the carrier) necessary for an AM SSB signal?

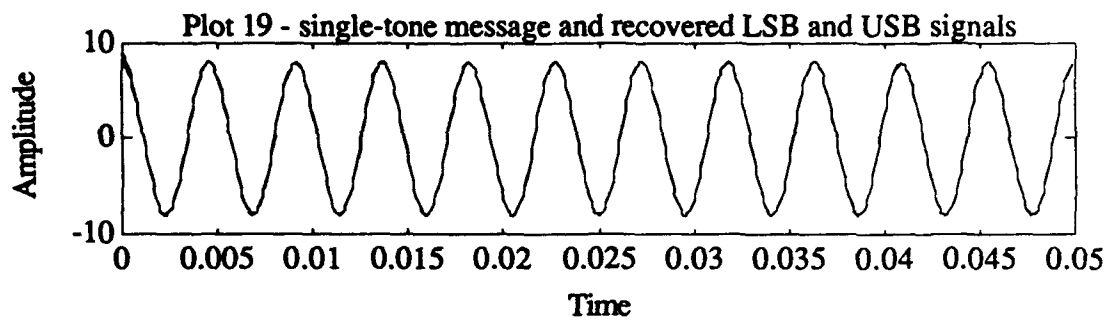
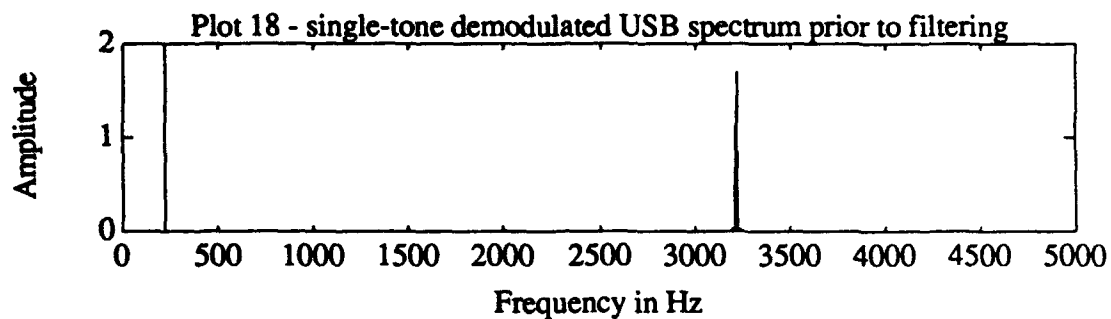
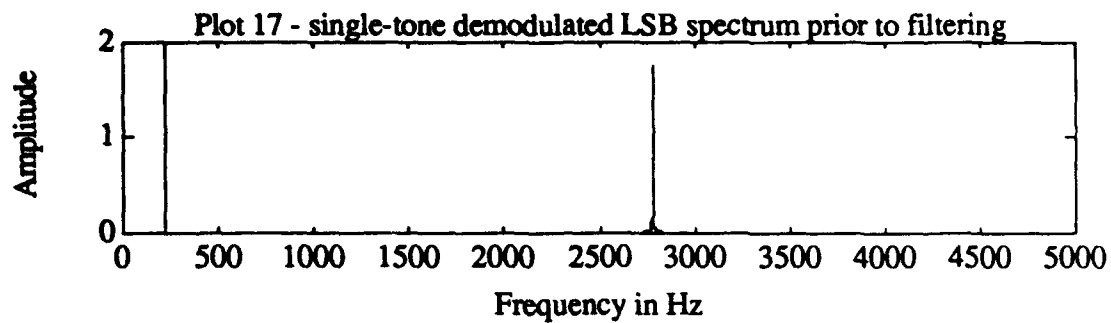
Answer: Yes, because although there are no phase shifts present in the AM SSB signals, their shapes do not closely follow the envelope of the message signal, preventing envelope detection.



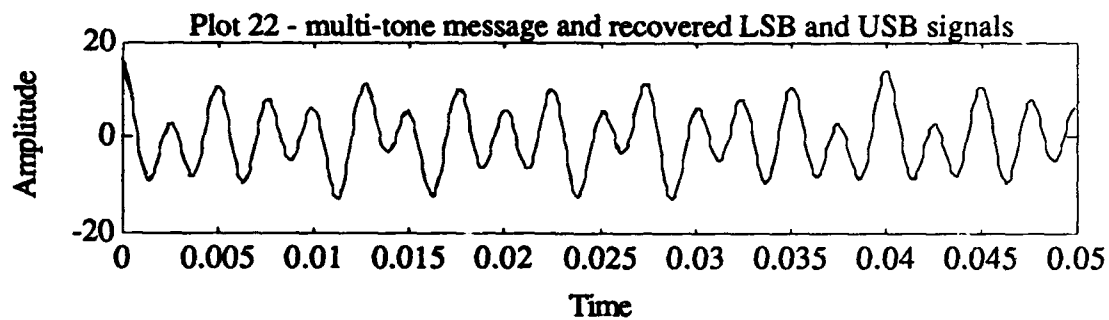
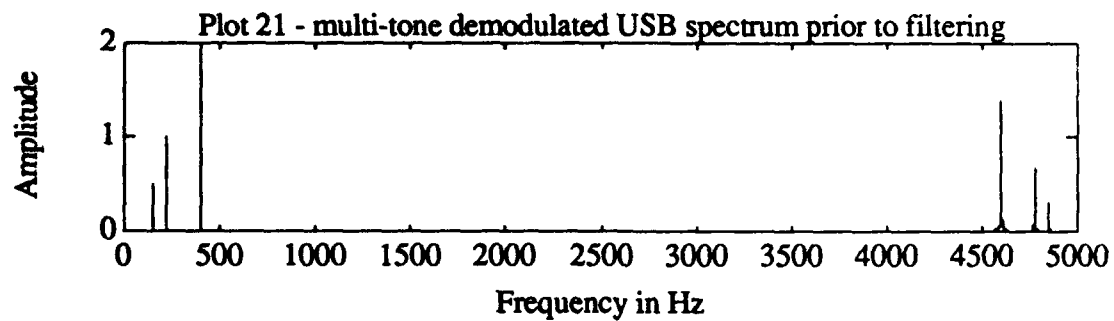
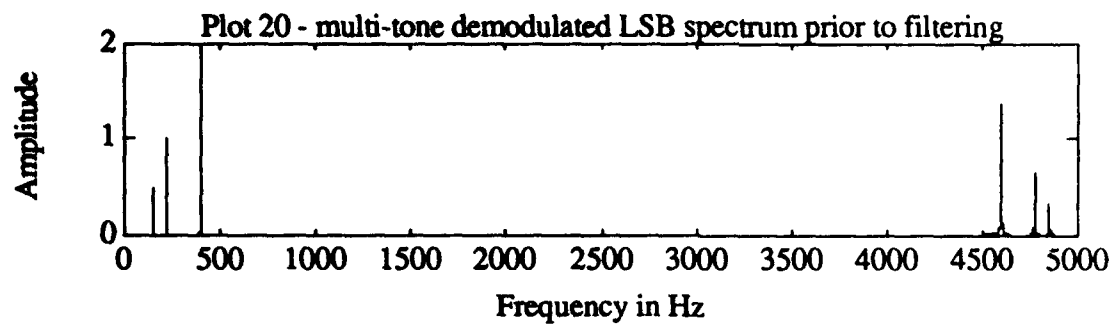








NO PLOT HERE--JUST PRESS RETURN



lab6_ex.m

%Programming Lab 6 example for instructor use

%Answers will vary!

%%

%Programming Lab 6 Amplitude Modulation AM SSB

%%

%Part 1--Generate single- and multi-tone message signals

% and spectra

%A. Generate the message signals

clear

clf

delta_t=.0001;

t=0:delta_t:1; %time vector

sgl=8*cos(2*pi*220*t); %single-tone signal

%multi-tone signal

mlt=2*cos(2*pi*150*t)+4*cos(2*pi*225*t)+8*cos(2*pi*400*t);

%Plot 1

subplot(211), %plot the signal

plot(t(1:500),sgl(1:500))

title('Plot 1 - single-tone message signal')

xlabel('Time')

ylabel('Amplitude')

%Plot 2

subplot(212), %plot the signal

plot(t(1:500),mlt(1:500))

title('Plot 2 - multi-tone message signal')

xlabel('Time')

ylabel('Amplitude')

pause

clf

%B. Predict the power and bandwidth for the message signals

%C. Verify bandwidth for the message signals

[specsgl,Hz]=spectral(sgl,delta_t); %generate the spectrum

%Plot 3

```

subplot(211), %plot the spectrum
plot(Hz,specsgl)
title('Plot 3 - single-tone message signal spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

specmlt=spectral(mlt,delta_t); %generate the spectrum

%Plot 4

subplot(212), %plot the spectrum
plot(Hz,specmlt)
title('Plot 4 - multi-tone message signal spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear specsgl;clear specmlt;

pause
clg

%%%%%%%%%%
%Part 2--Generate AM SSB signals using single- and multi-
%      tone input
%A. Generate the single- and multi-tone AM SSB signals

[lsbsgl,usbsgl]=ssb(t,sgl,1,1500); %generate the single-tone upper and lower
                                %sideband signals
[lsbmlt,usbmlt]=ssb(t,mlt,1,2500); %generate the multi-tone upper and lower
                                %sideband signals

%Plot 5

subplot(211), %plot the lower sideband signal
plot(t(1:500),lsbsgl(1:500));
title('Plot 5 - single-tone LSB signal')
xlabel('Time')
ylabel('Amplitude')

%Plot 6

subplot(212), %plot detailed view--no phase shifts
plot(u(1:100),[sgl(1:100);lsbsgl(1:100)])
title('Plot 6 - single-tone message signal, expanded LSB signal')
xlabel('Time')
ylabel('Amplitude')

pause

```

```

clg

%Plot 7

subplot(211), %plot the upper sideband signal
plot(t(1:500),usbsgl(1:500));
title('Plot 7 - single-tone USB signal')
xlabel('Time')
ylabel('Amplitude')

%Plot 8

subplot(212), %plot detailed view--no phase shifts
plot(t(1:100),[sgl(1:100);usbsgl(1:100)])
title('Plot 8 - single-tone message signal, expanded USB signal')
xlabel('Time')
ylabel('Amplitude')

pause
clg

%Plot 9

subplot(211), %plot the lower sideband signal
plot(t(1:500),lsbmlt(1:500));
title('Plot 9 - multi-tone LSB signal')
xlabel('Time')
ylabel('Amplitude')

%Plot 10

subplot(212), %plot detailed view--no phase shifts
plot(t(1:100),[mlt(1:100);lsbmlt(1:100)])
title('Plot 10 - multi-tone message signal, expanded LSB signal')
xlabel('Time')
ylabel('Amplitude')

pause
clg

%. . . 11

subplot(211), %plot the upper sideband signal
plot(t(1:500),usbmlt(1:500));
title('Plot 11 - multi-tone USB signal')
xlabel('Time')
ylabel('Amplitude')

```


%Plot 12

```
subplot(212), %plot detailed view--no phase shifts
plot(t(1:100),[mlt(1:100);usbmlt(1:100)])
title('Plot 12 - multi-tone message signal, expanded USB signal')
xlabel('Time')
ylabel('Amplitude')
```

```
pause
clg
```

%B. Predict power and bandwidth for the AM SSB signals

%C. Verify the power and bandwidth of the AM SSB signals

```
[speclsbsgl,HZ]=spectral(lsbgl,delta_t); %generate the lower sideband spectrum
[specusbsgl,HZ]=spectral(usbsgl,delta_t); %generate the upper sideband spectrum
```

%Plot 13

```
subplot(211), %plot the lower sideband spectrum
plot(HZ,speclsbsgl);
title('Plot 13 - single-tone LSB spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')
```

%Plot 14

```
subplot(212), %plot the upper sideband spectrum
plot(HZ,specusbsgl);
title('Plot 14 - single-tone USB spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')
```

```
clear specusbsgl;clear speclsbsgl;
```

```
pause
clg
```

```
[speclsbmlt,HZ]=spectral(lsbmlt,delta_t); %generate the lower sideband spectrum
[specusbmlt,HZ]=spectral(usbmlt,delta_t); %generate the upper sideband spectrum
```

%Plot 15

```
subplot(211), %plot the lower sideband spectrum
plot(HZ,speclsbmlt);
title('Plot 15 - multi-tone LSB spectrum')
xlabel('Frequency in Hz')
```

```

ylabel('Amplitude')

%Plot 16

subplot(212), %plot the upper sideband spectrum
plot(Hz,specusbmlt);
title('Plot 16 - multi-tone USB spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear specusbmlt;clear specsbsmlt;

pause
clg

lsb_pk_pwr_sngl=((max(lsbssl))^2)/2 %find the peak power
usb_pk_pwr_sngl=((max(usbsgl))^2)/2

[psdlsb,HZ]=psd(lsbssl,delta_t); %generate the lower sideband power
                                %spectral density
lsb_avg_pwr_sngl=sum(psdlsb) %find average power by summing the power
                                %spectral densities
[psdusb,HZ]=psd(usbsgl,delta_t); %generate the lower sideband power
                                %spectral density
lsb_avg_pwr_sngl=sum(psdusb) %find average power by summing the power
                                %spectral densities
clear psdusb;clear psdlsb;

%%%%%%%%%%%%%
%Part 3--Recover the AM SSB signals
%A. Recover the AM SSB single-tone signals

demodlsbssl=cos(2*pi*1500*t).*lsbssl; %recover the single-tone signal by
                                %multiplying by the carrier
demodusbssl=cos(2*pi*1500*t).*usbsgl;

clear lsbssl;clear usbsgl;

[specsbsgl,HZ,fftlsbssl]=spectral(demodlsbssl,delta_t);
clear demodlsbssl;

[specusbsgl,HZ,fftusbsgl]=spectral(demodusbssl,delta_t);
clear demodusbssl;

%Plot 17

subplot(211), %plot the demodulated lsb spectrum
plot(HZ,specsbsgl)

```

```

title('Plot 17 - single-tone demodulated LSB spectrum prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear specbsbgl;

%Plot 18

subplot(212), %plot the demodulated usb spectrum
plot(Hz,specusbsgl)
title('Plot 18 - single-tone demodulated USB spectrum prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear specusbsgl;

pause
clg

recbsbgl=recoverm(ffilsbsgl,'ideallow',Hz,250); %recover and filter
clear ffilsbsgl; %single-tone LSB
biglsbsgl=recbsbgl*4; %amplify signal
clear recbsbgl;

recusbsgl=recoverm(fftusbsgl,'ideallow',Hz,250); %recover and filter
clear fftusbsgl; %single-tone USB
bigusbsgl=recusbsgl*4; %amplify signal
clear recusbsgl;

%Plot 19

subplot(211), %plot the recovered signals over the message signal
plot(t(1:500),[sgl(1:500);biglsbsgl(1:500);bigusbsgl(1:500)])
title('Plot 19 - single-tone message and recovered LSB and USB signals')
xlabel('Time')
ylabel('Amplitude')

subplot(212),
title('NO PLOT HERE--JUST PRESS RETURN')

clear biglsbsgl;clear bigusbsgl;clear sgl;

pause
clg

%B. Recover the AM SSB multi-tone signals

demodlsbmlt=cos(2*pi*2500*t).*lsbmlt; %recover the multi-tone signal by

```

```

                                %multiplying by the carrier
demodusbmlt=cos(2*pi*2500*t).*usbmlt;

clear lsbmlt;clear usbmlt;

[speclsbmlt,HZ,fftlbmlt]=spectral(demodlsbmlt,delta_t);
clear demodlsbmlt;

[specusbmlt,HZ,fftusbmlt]=spectral(demodusbmlt,delta_t);
clear demodusbmlt;

%Plot 20

subplot(211), %plot the demodulated lsb spectrum
plot(HZ,speclsbmlt)
title('Plot 20 - multi-tone demodulated LSB spectrum prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear speclsbmlt;

%Plot 21

subplot(212), %plot the demodulated usb spectrum
plot(HZ,specusbmlt)
title('Plot 21 - multi-tone demodulated USB spectrum prior to filtering')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear specusbmlt;

pause
clg

reclsbmlt=recoverm(fftlbmlt,'ideallow',HZ,420); %recover and filter
clear fftlbmlt;                                %multi-tone LSB
biglsbmlt=reclsbmlt*4; %amplify signal
clear reclsbmlt;

recusbmlt=recoverm(fftusbmlt,'ideallow',HZ,420); %recover and filter
clear fftusbmlt;                                %multi-tone USB
bigusbmlt=recusbmlt*4; %amplify signal
clear recusbmlt;

%Plot 22

subplot(211), %plot the recovered signals over the message signal
plot((1:500),[mlt(1:500);biglsbmlt(1:500);bigusbmlt(1:500)])

```

```
title('Plot 22 - multi-tone message and recovered LSB and USB signals')  
xlabel('Time')  
ylabel('Amplitude')
```

EO 3513 Programming Laboratory 7 Key

Conventional Amplitude Modulation (Conventional AM)

Answers will vary. The answers below are based on the following signals:

$$s_{gl} = \cos(2\pi \cdot 150 \cdot t);$$

$$m_{lt} = 3 \cdot \cos(2\pi \cdot 430 \cdot t) + 2 \cdot \cos(2\pi \cdot 250 \cdot t) + \cos(2\pi \cdot 110 \cdot t);$$

Question 1: Calculate the following values for the single-tone message signal:

peak power
average power
baseband bandwidth

Answer: peak power = $P_p = \frac{A_p^2}{2} \Rightarrow 1^2 / 2 \Rightarrow 0.5$

average power = $P = A_0^2 + \frac{1}{2} \sum_{N=1}^{\infty} (A_N^2 + B_N^2) \Rightarrow 1^2 / 2 \Rightarrow 0.5$

baseband bandwidth = 150 Hz

Question 2: Predict the following values for the single-tone conventional AM signal:

peak power
average power
transmission bandwidth

Answer: peak power = $(1 + m)^2 P_c \Rightarrow (1 + 0.8)^2 \cdot 0.5 \Rightarrow 1.62$

average power = $P = \left(1 + \frac{m^2}{2}\right) P_c \Rightarrow (1 + (0.8^2 / 2)) \cdot 0.5 \Rightarrow 0.66$

transmission bandwidth = 300 Hz

Question 3: Record the values representing peak and average power for the signal-tone conventional AM signal.

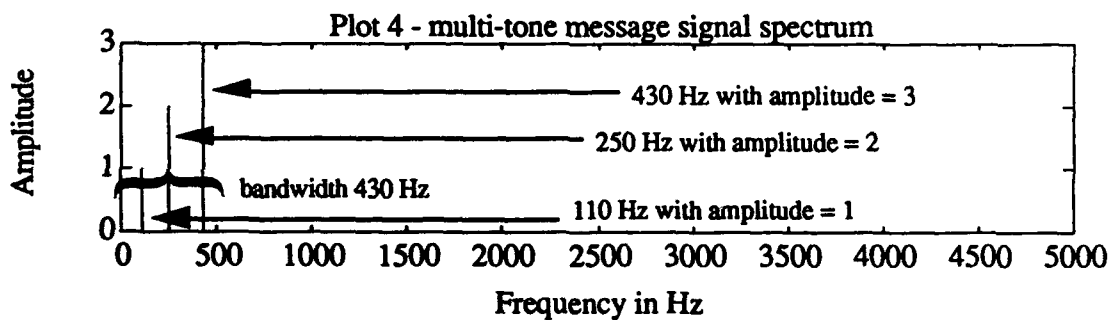
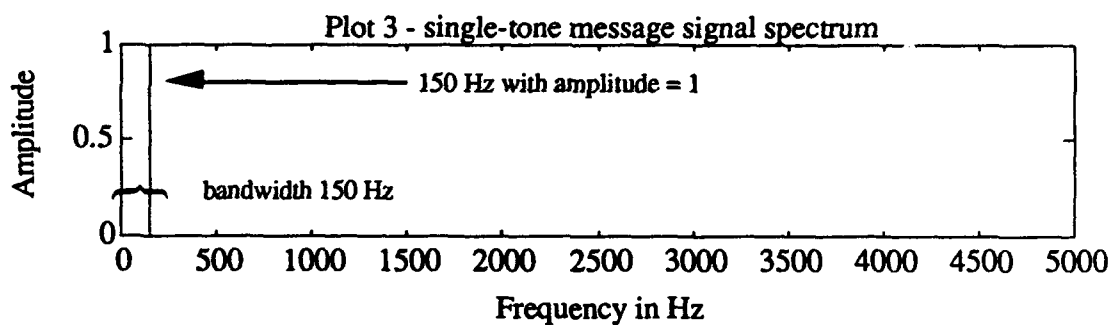
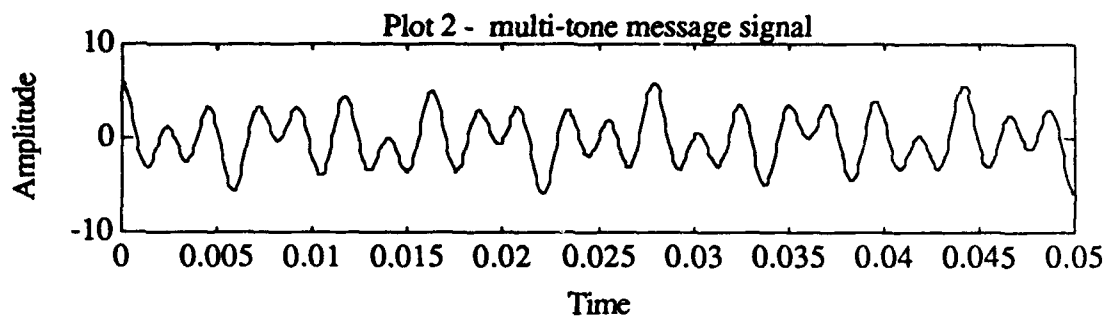
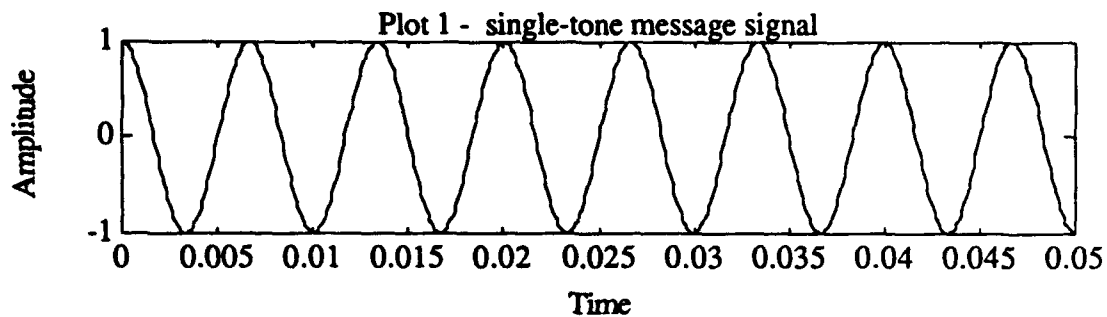
Do your calculations for bandwidth and power in Question 2 agree with the computer-generated values?

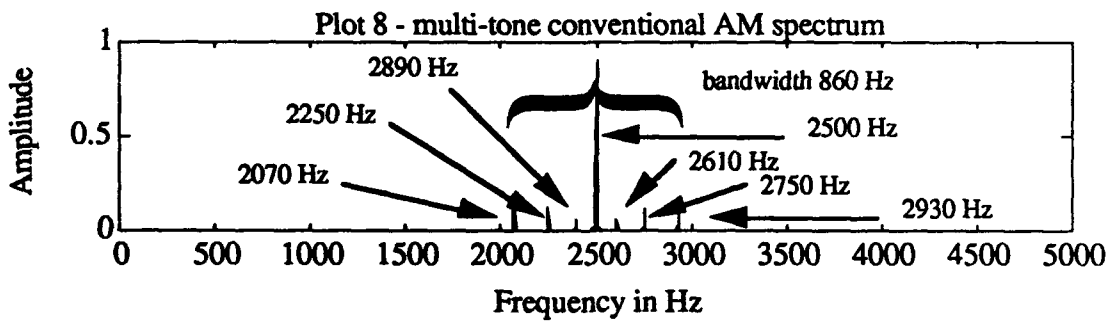
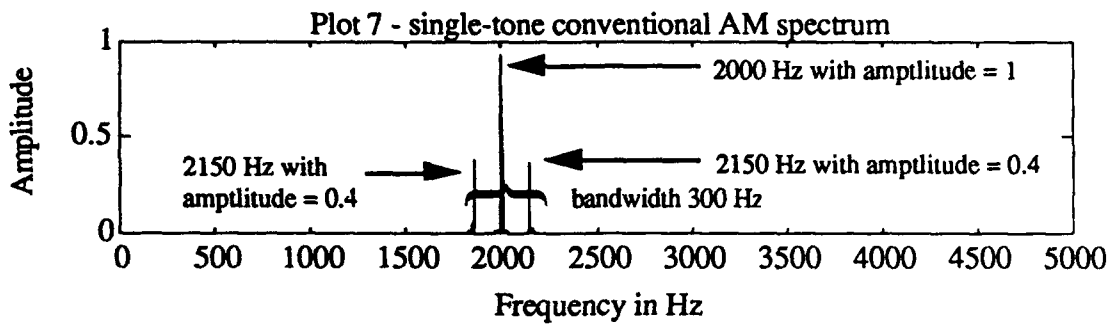
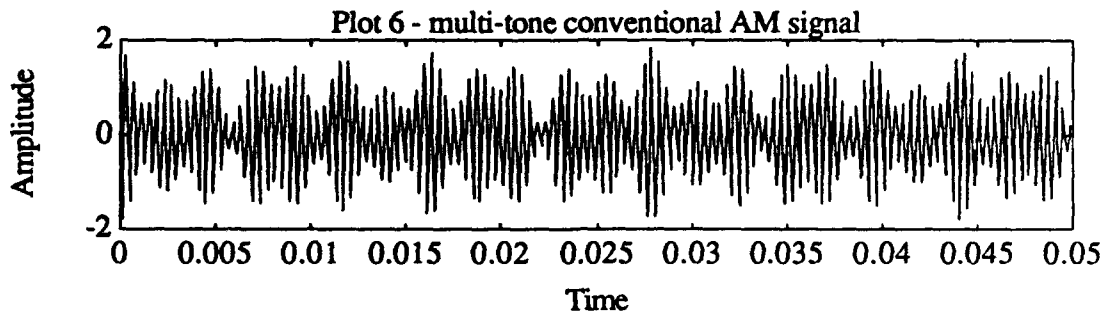
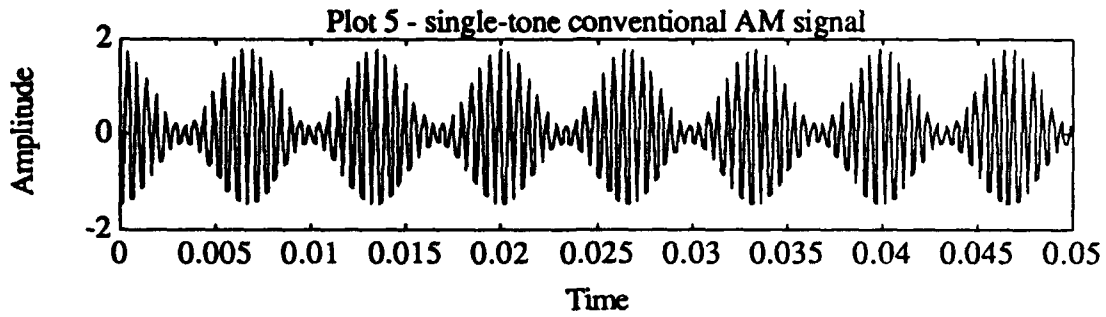
Answer: cam_pk_pwr_sgl = 1.6172
cam_avg_pwr_sgl = 0.6598

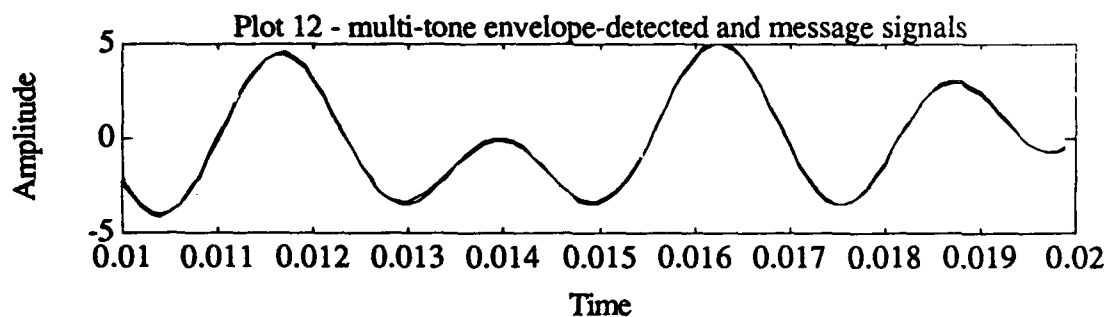
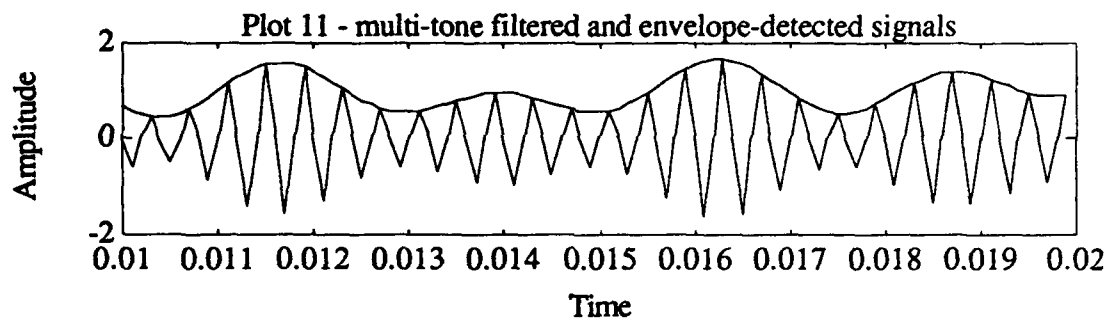
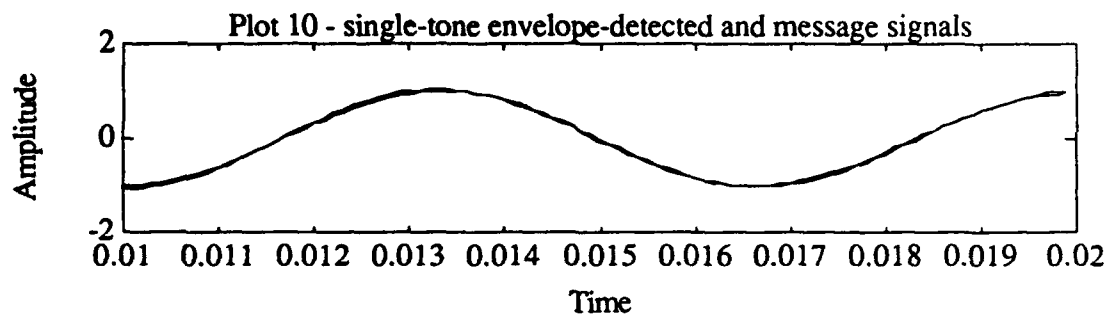
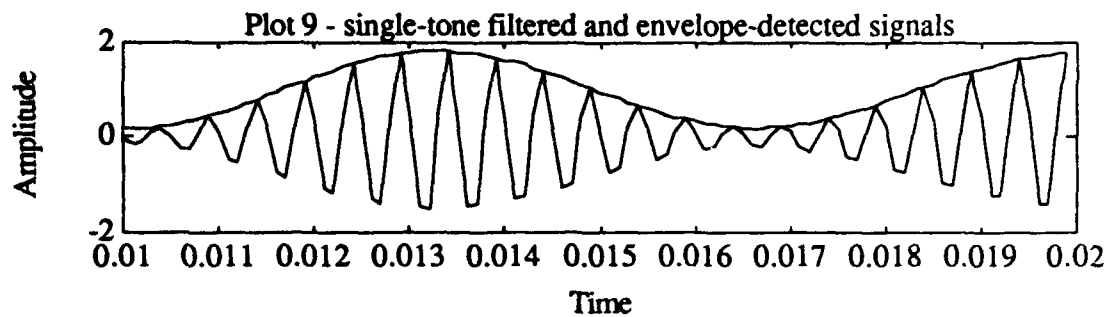
Yes—calculations agree.

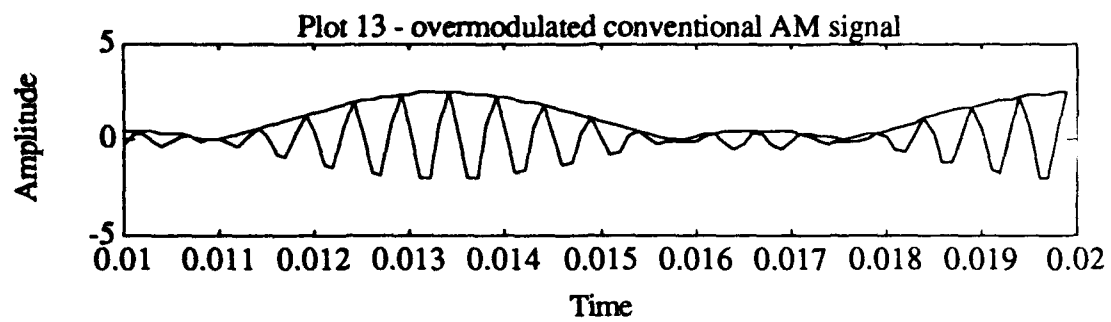
Question 4: What type of detection is needed for an overmodulated conventional AM signal? Why?

Answer: Coherent detection (detection using the carrier) is necessary for an overmodulated conventional AM signal. The phase shifts preclude envelope detection.









lab7_ex.m

%Lab 7 example script for instructor use

%%
%Programming Lab 7 Conventional Amplitude Modulation
% (Conventional AM)
%%

%PART 1--Generate single- and multi-tone message signals
% and spectra
%A. Generate the message signals

clear
clg

delta_t=.0001;
t=0:delta_t:1; %time vector
sgl=cos(2*pi*150*t); %single-tone signal
%multi-tone signal
mlt=3*cos(2*pi*430*t)+2*cos(2*pi*250*t)+cos(2*pi*110*t);
max_m=max(mlt); %save for amplifying detected signal

fc1=2000; %modulating frequencies for the carrier signals
fc2=2500;
fs1=150; %highest frequency in the single-tone message signal
fs2=430; %highest frequency in the multi-tone message signal
m=.8; %conventional AM modulation index
over_m=1.5; %index for overmodulated signal

%Plot 1

subplot(211), %plot the signal
plot(t(1:500),sgl(1:500))
title('Plot 1 - single-tone message signal')
xlabel('Time')
ylabel('Amplitude')

%Plot 2

subplot(212), %plot the signal
plot(t(1:500),mlt(1:500), 'b')
title('Plot 2 - multi-tone message signal')
xlabel('Time')
ylabel('Amplitude')

pause
clg

%B. Predict power and bandwidth for the message signals

%C. Verify bandwidth for the message signals

[specsgl,HZ]=spectral(sgl,delta_t); %generate the spectrum

%Plot 3

subplot(211), %plot the spectrum

plot(HZ,specsgl, 'g')

title('Plot 3 - single-tone message signal spectrum')

xlabel('Frequency in Hz')

ylabel('Amplitude')

[specmlt,HZ]=spectral(mlt,delta_t); %generate the spectrum

%Plot 4

subplot(212), %plot the spectrum

plot(HZ,specmlt)

title('Plot 4 - multi-tone message signal spectrum')

xlabel('Frequency in Hz')

ylabel('Amplitude')

clear specsgl;clear specmlt;

pause

clg

%%%

%PART 2--Generate conventional AM signals using single- and

% multi-tone input

%A. Generate the single- and multi-tone conventional AM

%signals

convamsgl=conv_am(sgl,delta_t,fc1,m);

convammlt=conv_am(mlt,delta_t,fc2,m);

sgl=sgl(1:300);

mlt=mlt(1:300);

%Plot 5

subplot(211), %plot the conventional AM modulated signal

plot(t(1:500),convamsgl(1:500))

title('Plot 5 - single-tone conventional AM signal')

xlabel('Time')

ylabel('Amplitude')

%Plot 6

```
subplot(212), %plot the conventional AM modulated signal
plot(t(1:500),convammlt(1:500), 'b')
title('Plot 6 - multi-tone conventional AM signal')
xlabel('Time')
ylabel('Amplitude')
```

```
pause
clg
```

%B. Predict power and bandwidth for the conventional AM signals

%C. Verify power and bandwidth for the conventional AM signals

```
                %generate the modulated spectrum
[camspecsgl,H,ffcamsgl]=spectral(convamsgl,delta_t);
[camspecmlt,H,ffcammlt]=spectral(convammlt,delta_t);
```

%Plot 7

```
subplot(211), %plot the modulated spectrum
plot(Hz,camspecsgl, 'g')
title('Plot 7 - single-tone conventional AM spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')
```

%Plot 8

```
subplot(212), %plot the modulated spectrum
plot(Hz,camspecmlt)
title('Plot 8 - multi-tone conventional AM spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')
```

```
clear camspecsgl;clear camspecmlt;clear convammlt;
```

```
pause
clg
```

```
cam_pk_pwr_sgl=((max(convamsgl))^2)/2 %find the peak power
```

```
psdcam=psd(convamsgl,delta_t); %generate the power spectral density
clear convamsgl;
cam_avg_pwr_sgl=sum(psdcam) %find average power by summing the power
                        %spectral density values
clear psdcam;
```

%%

%Part 3--Recover the conventional AM signals

%A. Recover the single-tone conventional AM signal

 %recover and filter

 filtsgl=recoverm(fftcamsgl,'idealbnd',Hz,1800,2200);

 filtsgl=filtsgl(1:300);

 clear fftcamsgl;

 envsgl=envelope(filtsgl);

 %plot the envelope-detected signal over the

%Plot 9 %filtered signal

 subplot(211),

 plot(t(101:200),[filtsgl(101:200);envsgl(101:200)])

 title('Plot 9 - single-tone filtered and envelope-detected signals')

 xlabel('Time')

 ylabel('Amplitude')

 bigsgl=(envsgl-1)/m; %remove DC value, divide by m

%Plot 10

 %plot the message signal

 subplot(212), %over the amplified envelope-detected signal

 plot(t(101:200),bigsgl(101:200))

 title('Plot 10 - single-tone envelope-detected and message signals')

 xlabel('Time')

 ylabel('Amplitude')

 hold on

 pause

 plot(t(101:200),sgl(101:200), 'b')

 hold off

 pause

 clg

clear bigsgl;clear filtsgl;

 filtm1t=recoverm(fftcamm1t,'idealbnd',Hz,2000,3000);

 clear fftcamm1t;

 filtm1t=filtm1t(1:300); %reduce length for speed

 envm1t=envelope(film1t); %use envelope detector

 %plot the envelope-detected

%Plot 11 %signal over the filtered signal

 subplot(211),

Programming Laboratory 7 Key—page 10

```

plot(t(101:200),[filtmlt(101:200);envmlt(101:200)])
title('Plot 11 - multi-tone filtered and envelope-detected signals')
xlabel('Time')
ylabel('Amplitude')

bigmlt=(envmlt-1)/m; %remove DC value, divide by m, amplify
biggermlt=bigmlt*max_m; %amplify the signal

%Plot 12
%message signal over
subplot(212), %envelope-detected signal
plot(t(101:200),biggermlt(101:200))
title('Plot 12 - multi-tone envelope-detected and message signals')
xlabel('Time')
ylabel('Amplitude')
hold on
pause
plot(t(101:200),mlt(101:200), 'b')
hold off

pause
clg

%%%%%%%%%%%%%%
%PART 4--Observe the effect of overmodulation on a single-tone
%conventional AM signal
%A. Overmodulate a single-tone conventional AM signal

oconvamsgl=conv_am(sgl,delta_t,fc1,over_m); %overmodulate the signal

%Plot 13

subplot(211), %plot the enlarged view of the overmodulated signal
plot(t(101:200),oconvamsgl(101:200))
title('Plot 13 - overmodulated conventional AM signal')
xlabel('Time')
ylabel('Amplitude')
hold on
pause

%B. Describe the effect of overmodulation on recovery

oenv=envelope(oconvamsgl); %use envelope detector

plot(t(101:200),oenv(101:200), 'b')
hold off

```


This page is intentionally
left blank.

EO 3513 Programming Laboratory 8 Key *Frequency Modulation (FM)*

Answers will vary. The answers below are based on the following signals:

$$sgl=15*\cos(2*\pi*50*t);$$

$$mlt=6*\cos(2*\pi*90*t)+9*\cos(2*\pi*30*t);$$

Question 1: Calculate the following values for the single-tone message signal:

peak power
average power
baseband bandwidth

Answer: peak power = $P_P = \frac{A_P^2}{2} \Rightarrow 15^2 / 2 \Rightarrow 112.5$

average power = $P = A_0^2 + \frac{1}{2} \sum_{N=1}^{\infty} (A_N^2 + B_N^2) \Rightarrow 15^2 / 2 \Rightarrow 112.5$

baseband bandwidth = 50 Hz

Question 2: Predict the following values for the single-tone FM signal:

peak power
average power
maximum frequency deviation Δf
transmission bandwidth (use Carson's rule)

Answer: peak power = $P_P = \frac{A_P^2}{2} \Rightarrow 15^2 / 2 \Rightarrow 112.5$

average power = $A^2 / 2 \Rightarrow 15^2 / 2 \Rightarrow 112.5$

$\Delta f = B * f_m \Rightarrow 10 * 50 \Rightarrow 500 \text{ Hz}$

transmission bandwidth = $2 B f_m \Rightarrow 2 * 10 * 50 \Rightarrow 1000 \text{ Hz}$

Question 3: Consult a table of values for Bessel functions (or use the MATLAB "bessel" command). How many sidebands are required for 98% power transmission for this FM signal? Does the spectrum shown in Plot 7 reflect the expected number of sidebands?

Answer: 14 sidebands are required.

The spectral plot reflects the expected number of sidebands.

Question 4: What is the distance between the sidebands in the FM spectrum shown in Plot 7?

Answer: 50 Hz

Question 5: Record the values representing peak and average power for the signal-tone FM signal.

Do your calculations for bandwidth and power in Question 2 agree with the computer-generated values?

Answer: $fm_pk_pwr_sngl = 112.5$
 $fm_avg_pwr_sngl = 112.4888$

Yes—calculations agrees.

Question 6: Calculate the maximum frequency deviation Δf associated with each of the two values of B .

Answer: $\Delta f = B * fm \Rightarrow 1 * 50 \Rightarrow 50 \text{ Hz}$
 $\Delta f = B * fm \Rightarrow 5 * 50 \Rightarrow 250 \text{ Hz}$

Question 7: Calculate the transmission bandwidth for each of the single-tone FM signals.

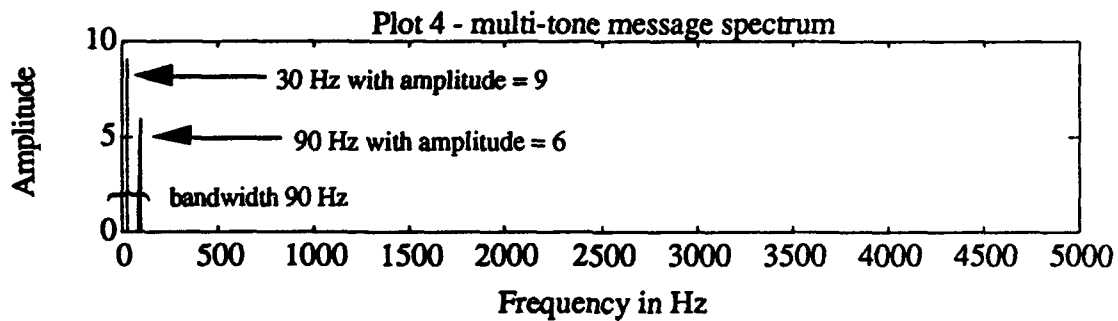
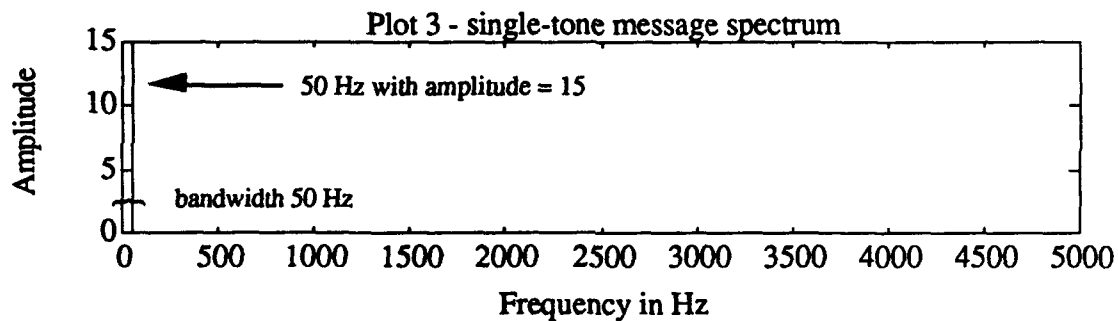
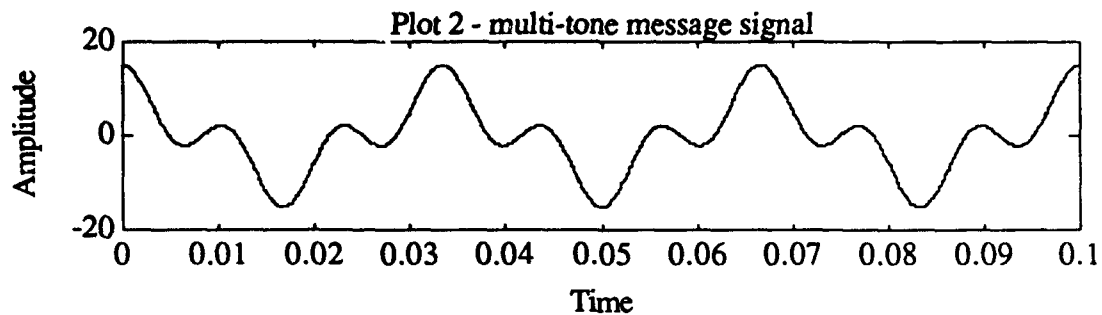
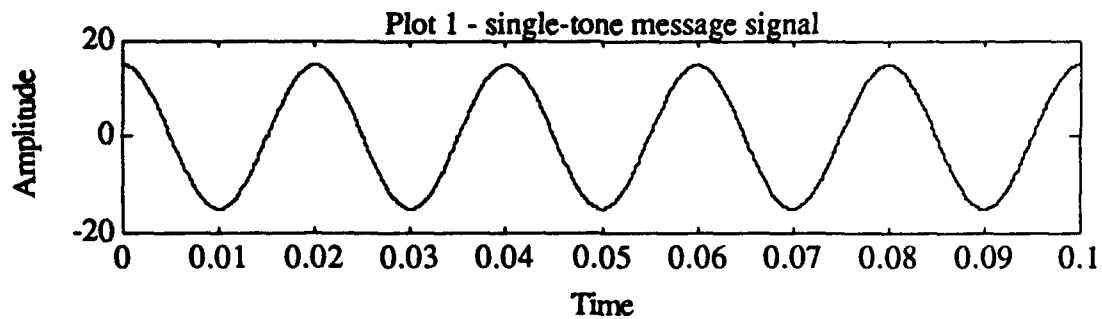
Answer: $B_T \approx 2 f_m \Rightarrow 2 * 50 \Rightarrow 100 \text{ Hz}$
 $B_T \approx 2 (1 + B) f_m \Rightarrow 2 (1 + 5) * 50 \Rightarrow 600 \text{ Hz}$

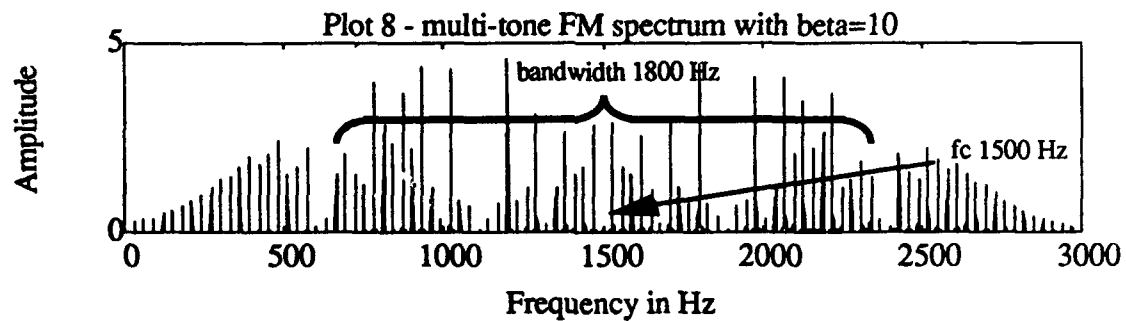
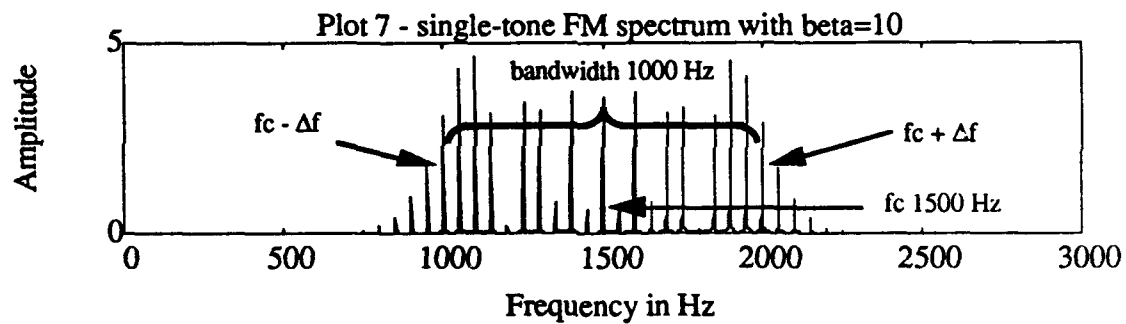
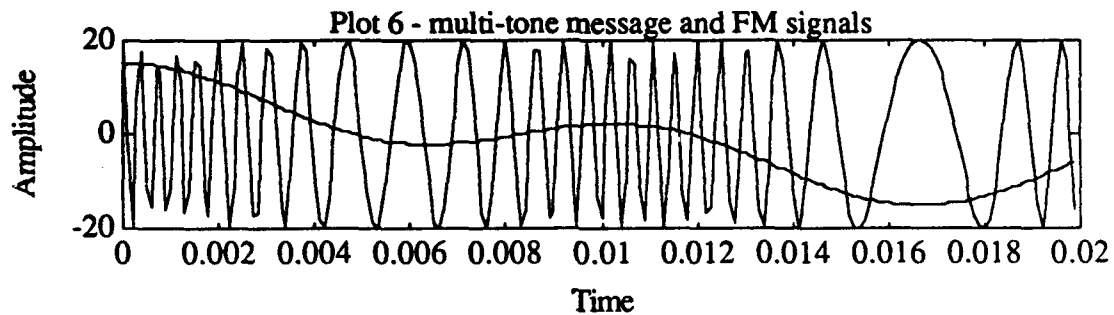
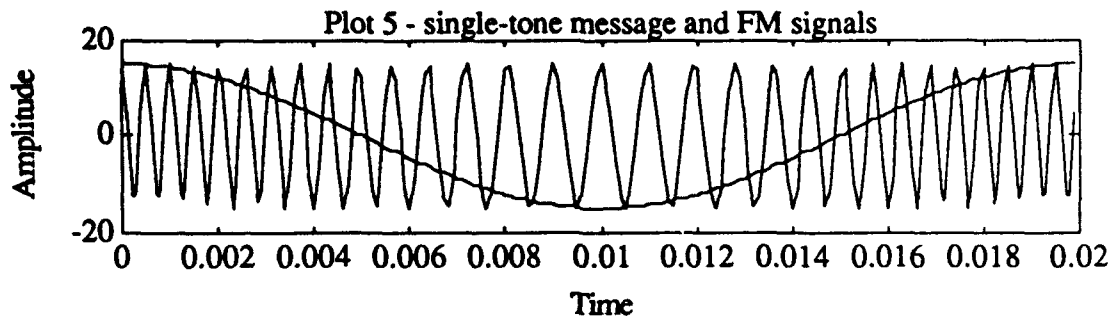
Question 8: Calculate the value of B associated with each of the two values of Δf .

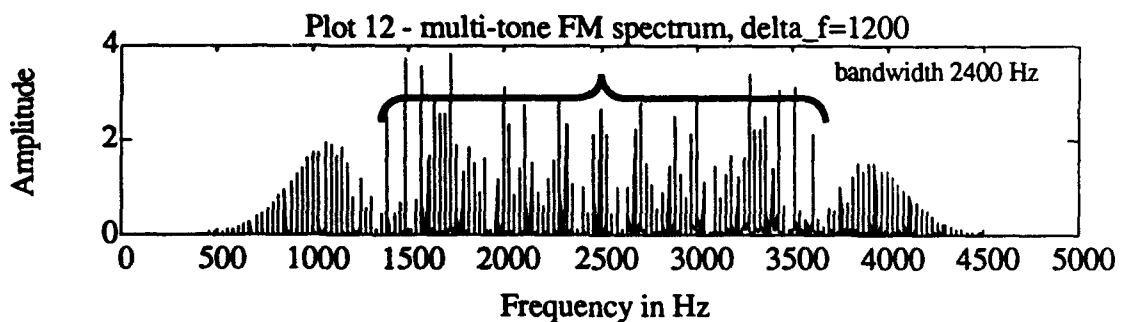
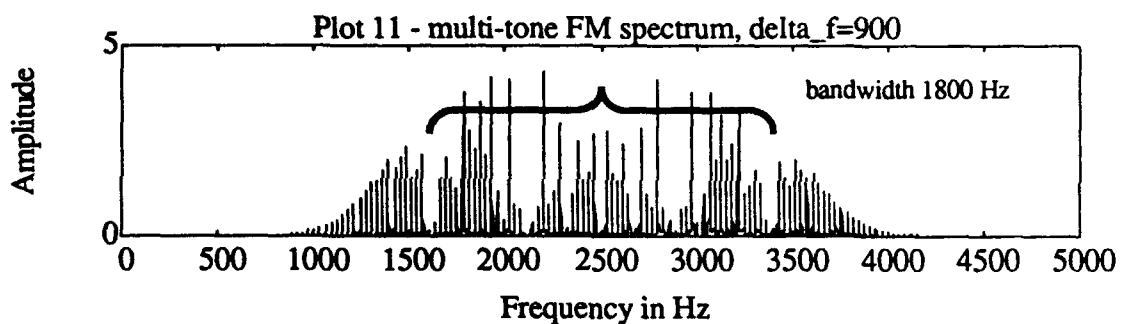
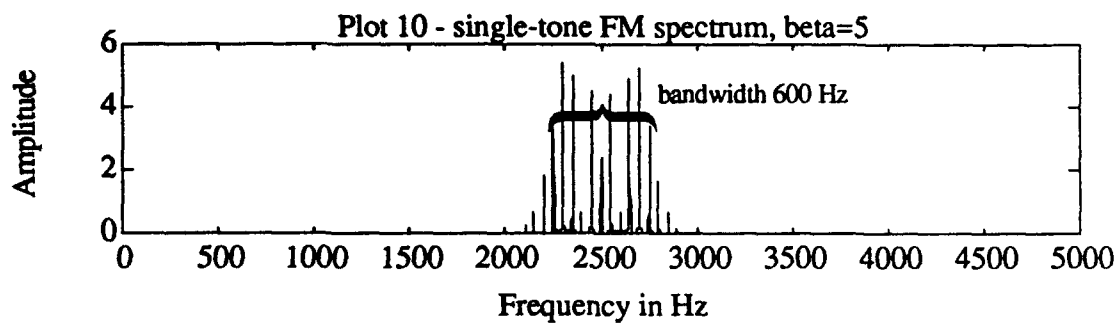
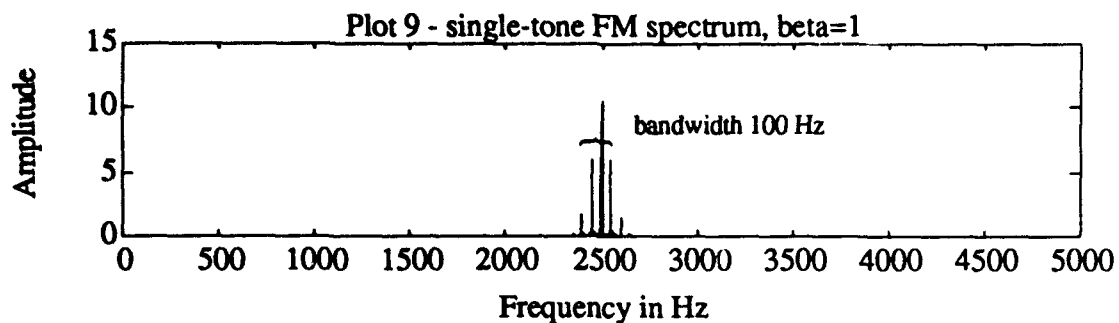
Answer: $B = \Delta f / fm \Rightarrow 900 / 90 \Rightarrow 10$
 $B = \Delta f / fm \Rightarrow 1200 / 90 \Rightarrow 13.333$

Question 9: Calculate the transmission bandwidth for each of the multi-tone FM signals (use the higher of the two values for message signal frequency).

Answer: $B_T \approx 2 B f_m \Rightarrow 2 * 10 * 90 \Rightarrow 1800 \text{ Hz}$
 $B_T \approx 2 B f_m \Rightarrow 2 * 13.333 * 90 \Rightarrow 2399.94 \text{ Hz}$







lab8_ex.m

%Lab 8 example script for instructor use
%Answers will vary!

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Programming Lab 8 Frequency Modulation (FM)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PART 1--Generate the single- and multi-tone message signals
%A. Generate the single- and multi-tone message signals

clear
clg

delta_t=.0001; %set signal and frequency variables
t=0:delta_t:1;

Acs=15; %FM signal amplitude for single-tone message
fcs=1500; %FM signal frequency for single-tone
thetas=0; %single-tone value
fms=50; %single-tone message signal frequency

sgl=15*cos(2*pi*fms*t); %single-tone signal

Acm=20; %FM signal amplitude for multi-tone message
fcm=1500; %FM signal frequency for multi-tone
thetam=[0 0]; %multi-tone value
freq1=90;
freq2=30;
fmm=[freq1 freq2]; %multi-tone frequency vector

mlt=6*cos(2*pi*freq1*t)+9*cos(2*pi*freq2*t); %multi-tone signal

%Plot 1

subplot(211),
plot(t(1:1000),sgl(1:1000))
title('Plot 1 - single-tone message signal')
xlabel('Time')
ylabel('Amplitude')

%Plot 2

subplot(212),
plot(t(1:1000),mlt(1:1000))
title('Plot 2 - multi-tone message signal')
xlabel('Time')
ylabel('Amplitude')
```



```
pause
clg
```

```
%B. Predict peak power, average power, and bandwidth for the
%message signals
```

```
%C. Generate the single- and multi-tone message signal spectra
```

```
[specsgl,HZ]=spectral(sgl,delta_t);
```

```
%Plot 3
```

```
subplot(211),
plot(HZ,specsgl)
title('Plot 3 - single-tone message spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')
```

```
[specmlt,HZ]=spectral(mlt,delta_t);
```

```
%Plot 4
```

```
subplot(212),
plot(HZ,specmlt)
title('Plot 4 - multi-tone message spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')
```

```
clear specsgl;clear specmlt;
```

```
pause
clg
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PART 2--Generate frequency modulated (FM) signals using
%single- and multi-tone input
%A. Generate the single- and multi-tone FM signals
```

```
beta=10;
```

```
%generate the FM signal
```

```
[fm_sgl,delta_fs,beta]=fm_mod(t,Acs,fcs,fms,thetas,'none',beta);
[fm_mlt,delta_fm,beta]=fm_mod(t,Acm,fcmm,fmm,thetam,'none',beta);
```

```
%Plot 5
```

```
subplot(211),
plot(t(1:200),[sgl(1:200);fm_sgl(1:200)])
```

Programming Laboratory 8 Key—page 8

```

title('Plot 5 - single-tone message and FM signals')
xlabel('Time')
ylabel('Amplitude')

%Plot 6

subplot(212),
plot(t(1:200),[m1t(1:200);fm_m1t(1:200)])
title('Plot 6 - multi-tone message and FM signals')
xlabel('Time')
ylabel('Amplitude')

pause
clg

%B Predict peak power, average power, and bandwidth
% for the FM signals

%C. Generate the spectra of the FM signals

[specsgl,HZ]=spectral(fm_sgl,delta_t);

%Plot 7

subplot(211), %label delta_f
plot(HZ(1:3000),specsgl(1:3000))
title('Plot 7 - single-tone FM spectrum with beta=10')
xlabel('Frequency in Hz')
ylabel('Amplitude')

[specm1t,HZ]=spectral(fm_m1t,delta_t);

%Plot 8

subplot(212), %label delta_f
plot(HZ(1:3000),specm1t(1:3000))
title('Plot 8 - multi-tone FM spectrum with beta=10')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

%D. Verify the power and bandwidth of the FM signal

psdfm=psd(fm_sgl,delta_t);

fm_pk_pwr_sngl=(max(fm_sgl)^2)/2

```

```

fm_avg_pwr_sngl=sum(psd_fm)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PART 3--Control the bandwidth of the FM signal
%A. Control bandwidth by varying beta, using single-tone input
    %fix beta at values of 1 and 5
    %modulate at 2500 Hz to center the spectrum

fcs=2500;
beta=1;

[fm_sgl,delta_f,beta]=fm_mod(t,Acs,fcs,fms,thetas,'none',beta);
delta_f
fm_sgl=spectral(fm_sgl,delta_t);

%Plot 9

subplot(211),
plot(Hz,fm_sgl)
title('Plot 9 - single-tone FM spectrum, beta=1')
xlabel('Frequency in Hz')
ylabel('Amplitude')

beta=5;

[fm_sgl,delta_f,beta]=fm_mod(t,Acs,fcs,fms,thetas,'none',beta);
delta_f
fm_sgl=spectral(fm_sgl,delta_t);

%Plot 10

subplot(212),
plot(Hz,fm_sgl)
title('Plot 10 - single-tone FM spectrum, beta=5')
xlabel('Frequency in Hz')
ylabel('Amplitude')

pause
clg

%B. Control bandwidth of the FM signal by varying delta_f,
% using multi-tone input
    %fix delta_f at values of 900 and 1200
    %Modulate at 2500 Hz to center the spectrum

fcm=2500;
delta_fm=900;

[fm_mlt,delta_f,beta]=fm_mod(t,Acm,fcm,fmm,thetam,delta_fm,'none');

```

```

beta
specmlt=spectral(fm_mlt,delta_t);

%Plot 11

subplot(211),
plot(Hz,specmlt)
title('Plot 11 - multi-tone FM spectrum, delta_f=900')
xlabel('Frequency in Hz')
ylabel('Amplitude')

delta_fm=1200;

[fm_mlt,delta_f,beta]=fm_mod(t,Acm,fc,fmm,thetam,delta_fm,'none');
beta
specmlt=spectral(fm_mlt,delta_t);

%Plot 12

subplot(212),
plot(Hz,specmlt)
title('Plot 12 - multi-tone FM spectrum, delta_f=1200')
xlabel('Frequency in Hz')
ylabel('Amplitude')

```

**This page is intentionally
left blank.**

EO 3513 Programming Laboratory 9 Key
Radio Frequency Digital Modulation Methods
(ASK, FSK, BPSK, and QPSK)

Answers will vary.

Question 1: Calculate the bit duration τ for this signal.

Answer: bit duration = $1/\text{bit rate} \Rightarrow 1/100 \Rightarrow 0.01$ seconds

Question 2: Calculate the approximate baseband bandwidth of the NRZL unipolar digital message signal.

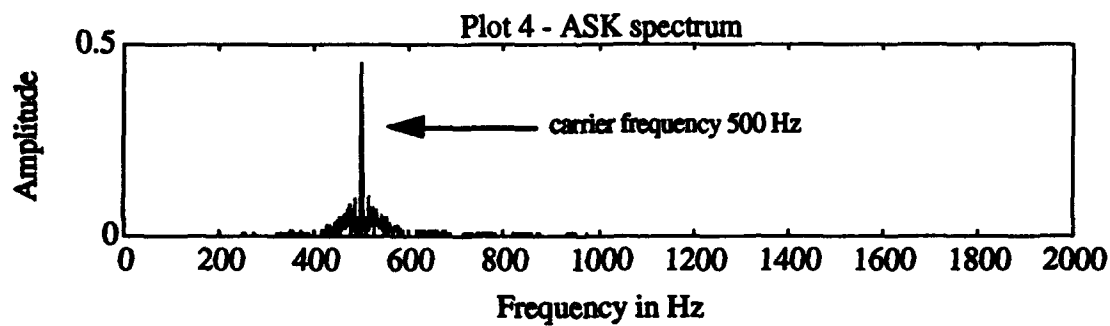
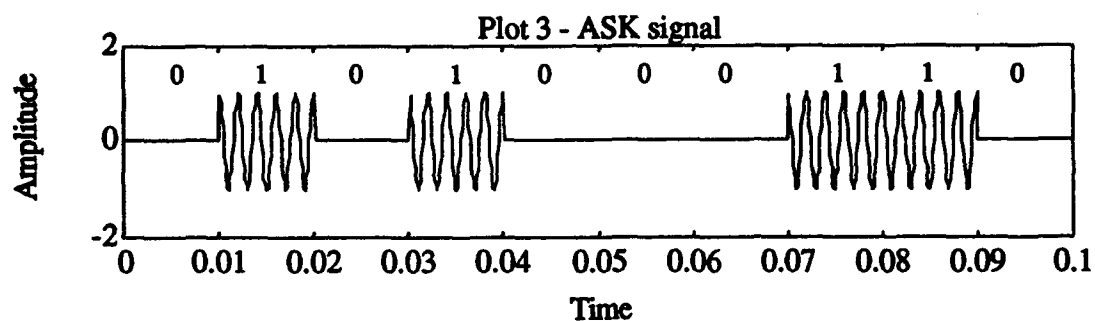
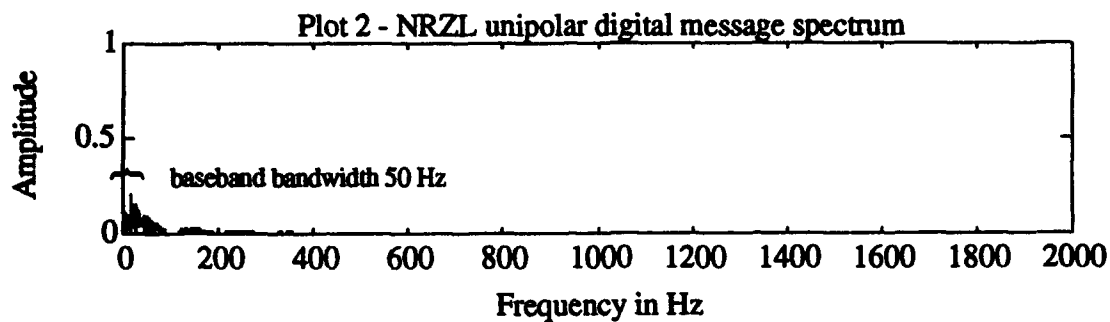
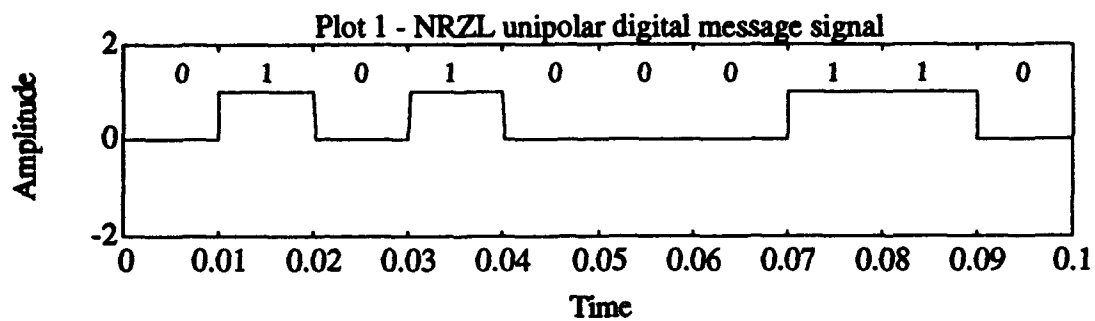
Answer: baseband bandwidth = $0.5/\tau \Rightarrow 0.5/0.01 \Rightarrow 50$ Hz

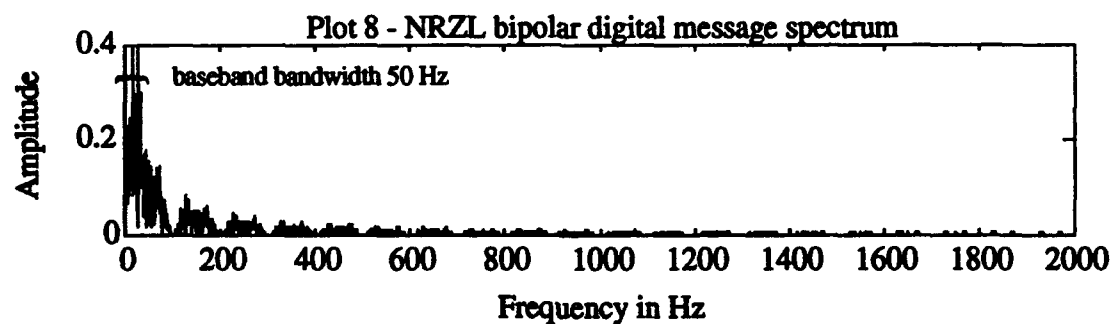
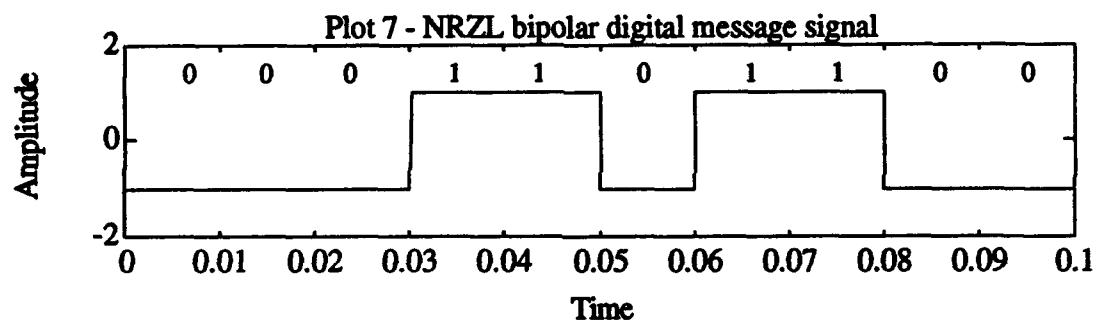
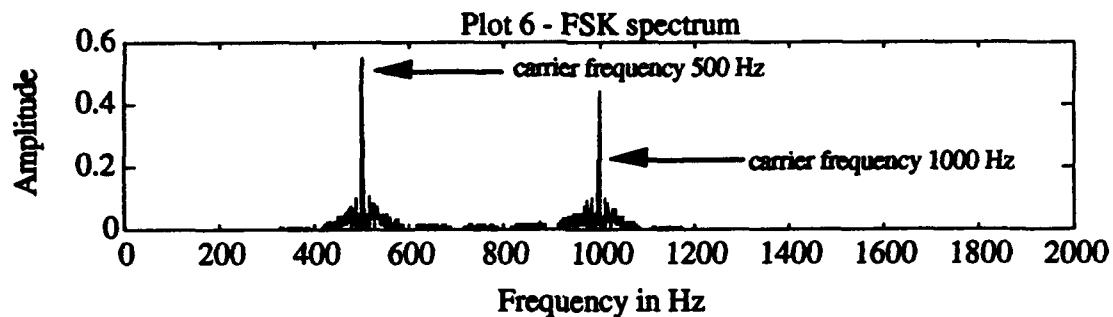
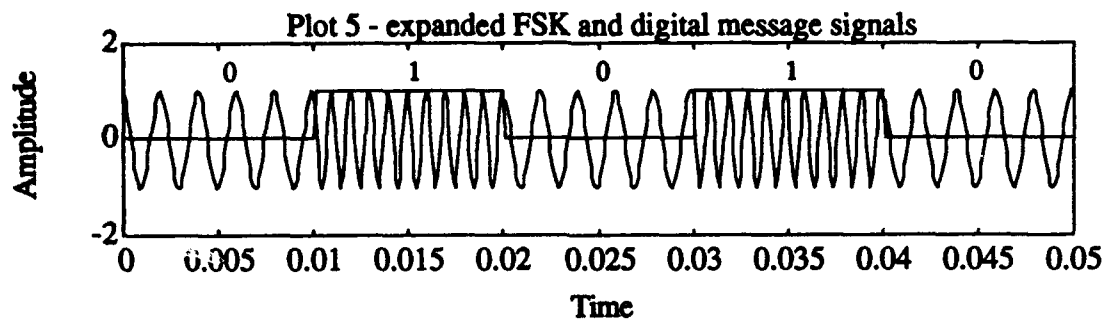
Question 3: Why is ASK modulation often referred to as “on-off keying”?

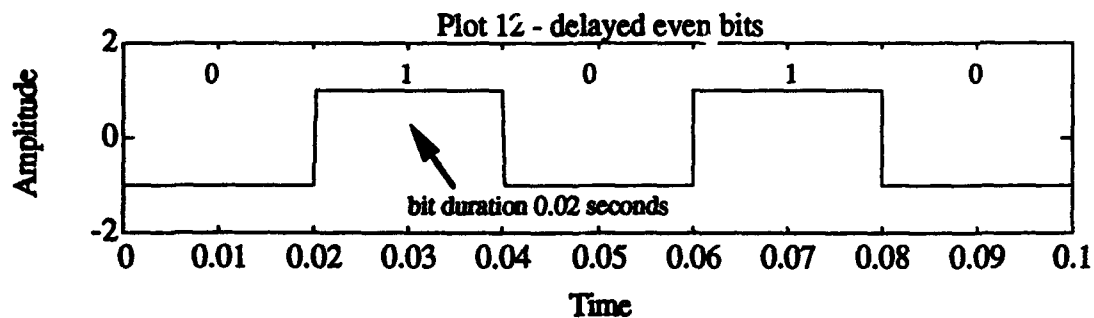
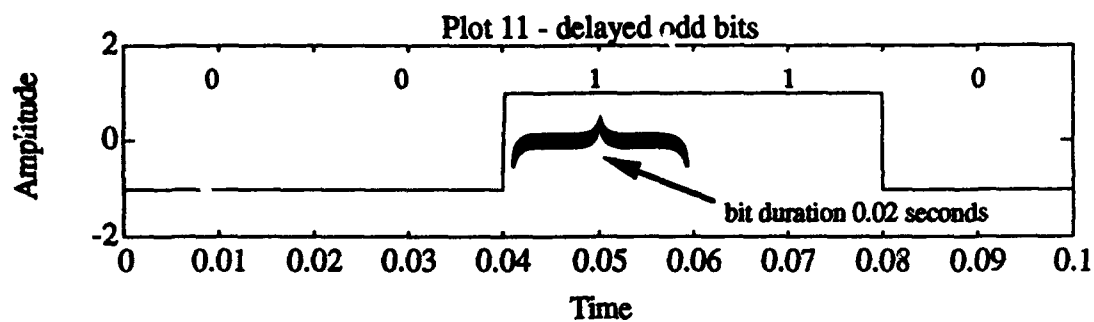
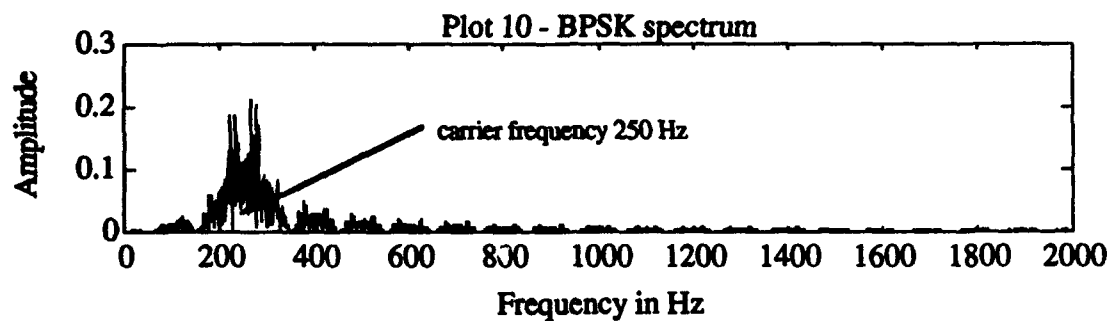
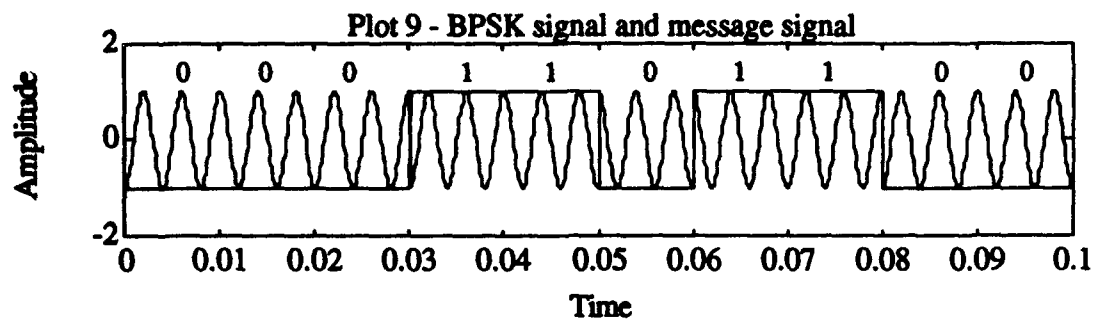
Answer: The carrier is turned “on” and “off” to represent the 1’s and 0’s in the digital message signal.

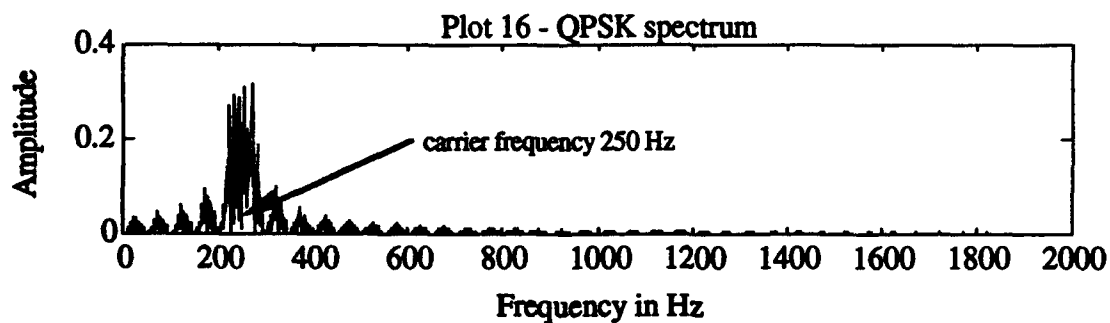
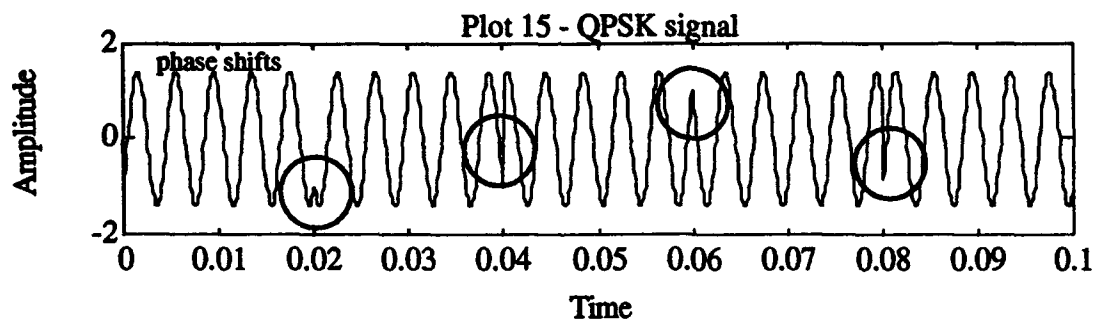
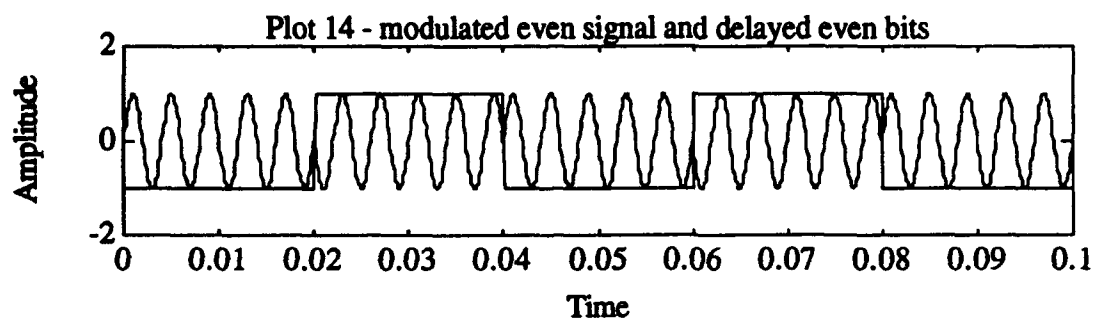
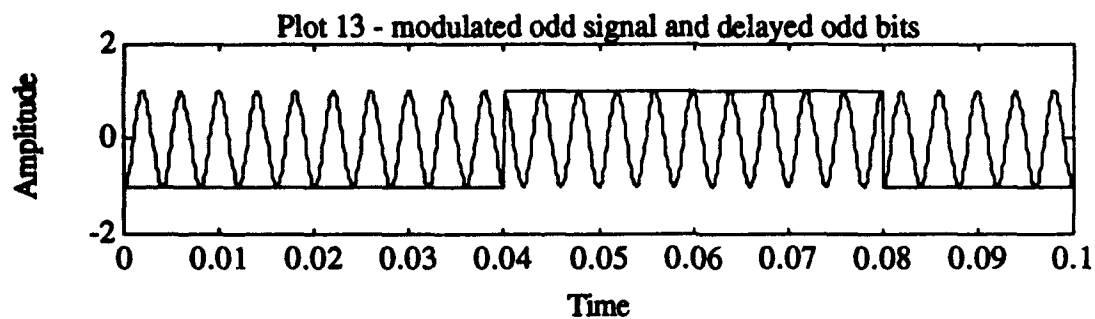
Question 4: Compare the spectrum for the BPSK signal in Plot 10 with that of the QPSK signal in Plot 16. What is the chief advantage of quadriphase shift keying over bipolar phase shift keying?

Answer: The information rate of a QPSK signal is twice that of a BPSK signal, with no increase in bandwidth requirements.









lab9_ex.m

%Example Lab 9 script for instructor use

```
%%%%%%%%%%%%%%
%Programming Laboratory 9 Radio Frequency (RF)
%           Digital Modulation
%%%%%%%%%%%%%%
%PART 1--Amplitude Shift Keying (ASK)
%A. Generate the digital message signal and spectrum

clear
clg

delta_t=.0001;
fc=500;
bitrate=100;

bitstream=round(rand(1:100)); %generate the random bitstream
unipolar_bits=bitstream(1:10) %print the first 10 bits
pause

[nrzsigt,t]=nrzlsig(bitstream,delta_t,bitrate); %generate the digital signal

big_axis=[0 .1 -2 2]; %set manual scaling for graphs
med_axis=[0 .05 -2 2];

axis(big_axis);

%Plot 1

subplot(211), %plot the digital message signal
plot(t,nrzsigt)
title('Plot 1 - NRZL unipolar digital message signal')
xlabel('Time')
ylabel('Amplitude')

[nrzlpec,HZ]=spectral(nrzsigt,delta_t); %generate the message spectrum

axis;

%Plot 2

subplot(212),
plot(HZ(1:2000),nrzlpec(1:2000), 'g')
title('Plot 2 - NRZL unipolar digital message spectrum')
xlabel('Frequency in Hz')
```

```

ylabel('Amplitude')

clear nrzlspec;

pause
clg

%B. Generate the ASK signal

asksig=nrzlsig.*cos(2*pi*fc*t); %generate the ASK signal

axis(big_axis);

%Plot 3

subplot(211), %plot the ASK signal
plot(t,asksig)
title('Plot 3 - ASK signal')
xlabel('Time')
ylabel('Amplitude')

%C. Generate the ASK spectrum

askspec=spectral(asksig,delta_t); %generate the ASK spectrum
clear asksig;

axis;

%Plot 4

subplot(212),
plot(Hz(1:2000),askspec(1:2000), 'b')
title('Plot 4 - ASK spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear askspec;

pause
clg

%%%%%%%%%%%%%%
%PART 2--Frequency Shift Keying (FSK)
%A. Generate the FSK signal

low_freq=500;
hi_freq=1000;

```

```
fsksig=fsk(nrzlsig,delta_t,bitrate,low_freq,hi_freq); %generate the FSK signal
```

```
axis(med_axis); %manually scale graph
```

```
%Plot 5
```

```
subplot(211), %FSK signal plotted over message signal  
plot(t,[nrzlsig;fsksig])  
title('Plot 5 - expanded FSK and digital message signals')  
xlabel('Time')  
ylabel('Amplitude')
```

```
%B. Generate the FSK spectrum
```

```
[fskspec,HZ]=spectral(fsksig,delta_t); %observe the FSK spectrum  
clear fsksig;
```

```
axis;
```

```
%Plot 6
```

```
subplot(212),  
plot(HZ(1:2000),fskspec(1:2000))  
title('Plot 6 - FSK spectrum')  
xlabel('Frequency in Hz')  
ylabel('Amplitude')
```

```
clear fskspec;
```

```
pause  
clg
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%PART 3--Binary Phase Shift Keying (BPSK)
```

```
%A. Generate the digital message signal and spectrum
```

```
%set bit pattern to show phase shifts at beginning
```

```
bitstream(1:10)=[0 0 0 1 1 0 1 1 0 0]; %set values to  
%demonstrate phase shifts
```

```
bipolar_bits=bitstream(1:10)
```

```
pause  
fc=250;
```

```
[nrzlsig,t]=nrzlbj(bitstream,delta_t,bitrate); %generate the digital  
%message signal
```

```
axis(big_axis);
```

```
%Plot 7
```

```

subplot(211), %plot the message signal
plot(t,nrzlsig)
title('Plot 7 - NRZL bipolar digital message signal')
xlabel('Time')
ylabel('Amplitude')

[nrzlsec,HZ]=spectral(nrzlsig,delta_t); %generate the message spectrum

axis;

%Plot 8

subplot(212),
plot(HZ(1:2000),nrzlsec(1:2000), 'g')
title('Plot 8 - NRZL bipolar digital message spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')

clear nrzlsec;

pause
clg

%B. Generate the BPSK signal

bpsksig=nrzlsig.*cos(2*pi*fc.*t); %generate the BPSK signal

axis(big_axis); %set manual scaling

%Plot 9

subplot(211), %plot the BPSK signal over the message signal
plot(t,[bpsksig,nrzlsig])
title('Plot 9 - BPSK signal and message signal')
xlabel('Time')
ylabel('Amplitude')

%C. Generate the BPSK spectrum

[bpskspec,HZ]=spectral(bpsksig,delta_t);
clear bpsksig;

axis;

%Plot 10

subplot(212), %plot the BPSK spectrum

```

```

plot(Hz(1:2000),bpskspec(1:2000), 'g')
title('Plot 10 - BPSK spectrum')
xlabel('Frequency in Hz')
ylabel('Amplitude')
clear bpskspec;

pause
clg

%%%%%%%%%%%%%%
%PART 4--Quadrphase Shift Keying
%A. Generate the QPSK signal

[nrzlodd,nrzleven]=ser_par(nrzlsig,delta_t,bitrate); %put signal through
                                                    %serial-to-parallel
                                                    %converter

axis(big_axis);

%Plot 11

subplot(211),      %graph the delayed digital signal--odd bits
plot(t,nrzlodd, 'b')
title('Plot 11 - delayed odd bits')
xlabel('Time')
ylabel('Amplitude')

%Plot 12

subplot(212),      %graph the delayed digital signal--even bits
plot(t,nrzleven, 'g')
title('Plot 12 - delayed even bits')
xlabel('Time')
ylabel('Amplitude')

pause
clg

cos_mod=nrzlodd.*cos(2*pi*fc.*t); %modulate each signal
sin_mod=nrzleven.*(-sin(2*pi*fc.*t));

%Plot 13

subplot(211),
plot(t,[cos_mod;nrzlodd])
title('Plot 13 - modulated odd signal and delayed odd bits')
xlabel('Time')
ylabel('Amplitude')

```

%Plot 14

```
subplot(212),  
plot(t,[sin_mod;nrzleven])  
title('Plot 14 - modulated even signal and delayed even bits')  
xlabel('Time')  
ylabel('Amplitude')  
pause  
cig
```

```
qpsk_sig=cos_mod+sin_mod; %sum the signals in the time domain  
clear cos_mod;clear sin_mod;
```

%Plot 15

```
subplot(211),  
plot(t,qpsk_sig, 'b')  
title('Plot 15 - QPSK signal')  
xlabel('Time')  
ylabel('Amplitude')
```

```
[qpskspec,HZ]=spectral(qpsk_sig,delta_t); %generate the QPSK spectrum  
clear qpsk_sig;
```

```
axis;
```

%Plot 16

```
subplot(212),  
plot(HZ(1:2000),qpskspec(1:2000))  
title('Plot 16 - QPSK spectrum')  
xlabel('Frequency in Hz')  
ylabel('Amplitude')
```


**This page is intentionally
left blank.**

APPENDIX E—COMMUNICATIONS TOOLBOX FOR MATLAB

compress.m

```
%function comsig=compress(s,mu,Vm)
%
%COMPRESS compresses the signal 's' using using the mu-255
%companding law, returning 'comsig'. Values of mu must be
%between 1 and 255; mu = 1 results in a linear function
%(no compression).
%Input parameters:
% s--the signal for compression and later expansion
% mu--the degree of compression effected (mu = 255 delivers the
%   highest compression, mu = 1 results in no compression)
% Vm--the maximum voltage in the signal. (Note: This value
%   must be exact! Using the 'max' command to pass in the
%   variable ensures its accuracy.)

%Written by Mike Shields 27 Jul 93
%Edited by Susan Guckelberg 5 Dec 93

function comsig=compress(s,mu,Vm);

signs=(2*sign(s))-1; %find the sign of each value of s and adjust to
                    %avoid division by 0
b=log(mu.*(abs(s)/Vm)+1); %the compression function
b=b.*(Vm/log(1+mu));

comsig=b.*signs; %restore the signs to s
```

convert.m

```
%function conv_num=convert(decimal,symbols,elements)
%
%CONVERT takes a decimal number value 'decimal' and returns its
%representation using the number of 'symbols' raised
%to the number of 'elements'.
%Input parameters:
% decimal--a base 10 number for conversion to another base
% symbols--the number of symbols in the coding scheme
% (binary=2, quad=4, hex=16)
% elements--the number of places in the coding scheme
% (the power to which the number of symbols is raised)

%Written by Susan Guckelberg 25 Jul 93

function conv_num=convert(decimal,symbols,elements);

for k=1:elements
    conv_num(elements+1-k)=rem(decimal,symbols);
    decimal=fix(decimal/symbols);
end
```

conv_am.m

```
%function convamsig=conv_am(msg,delta_t,fc,m)
%
%CON_AM normalizes the input message signal 'msg',
%raises it by 1, modulates it using 'm' and 'fc', and returns the
%conventional AM modulated signal 'convamsig'.
%Input parameters:
% msg--the message signal, normalized
% delta_t--the step size of the time vector
% fc--the frequency of the carrier (modulating) signal
% m--the index of modulation (values between 0 and 1)

%Written by Susan Guckelberg 10 Aug 93

function convamsig=conv_am(msg,delta_t,fc,m);

t=(1:length(msg))*delta_t; %regenerate the time vector

normsig=msg./max(abs(msg)); %normalize the signal

convamsig=(1+m*normsig).*cos(2*pi*fc*t); %multiply by the modulation index,
                                         %add 1 to raise the values above 0,
                                         %and modulate the signal
```

encode.m

```
%function codedsig=encode(bin_nums,symbols,elements)
%
%ENCODE accepts the vector of bin numbers 'bin_nums'
%generated by 'quantize.m'. For each number in
%'bin_nums', convert.m is called to
%convert it to the desired base ('symbols'). It returns
%the encoded signal 'codedsig', the length of bin_nums
%multiplied by 'elements'.
%Input parameters:
% bin_nums--a vector of bin numbers returned from bipolar.m
% symbols--the number of symbols in the coding scheme
% (binary=2, quad=4, hex=16)
% elements--the number of places in the coding scheme
% (the power to which the number of symbols is raised)

%Written by Susan Guckelberg 24 Jul 93

function codedsig=encode(bin_nums,symbols,elements);

codedsig=[];

for i=1:length(bin_nums)
    codednum=convert(bin_nums(i),symbols,elements);
    codedsig=[codedsig codednum];
end
```

envelope.m

```
%function envsig=envelope(x)
%
%ENVELOPE performs an envelope detection on the input signal
%'x' by computing the Hilbert transform of x, resulting in
%the magnitude of the complex envelope.
%Input parameter:
% x--the signal to be envelope-detected
```

```
%Written by Dennis W. Brown 8 Sep 93
%Edited by Susan Guckelberg 26 Dec 93
%Hilbert transform section written by Charles Denham 7 Jan 88
%Revised by LS, 19 Nov 88, 22 May 90, TPK 4 Nov 92
%Copyrighted by The MathWorks, Inc. 1988,1990,1992
```

```
function envsig=envelope(x);
```

```
envsig=[];
```

```
%the following section is based on the function
%HILBERT copyrighted by The MathWorks, Inc.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[r,c]=size(x); %find the size of x
if r==1
    x=x.'; %transpose the vector into a column
end
```

```
[n,cc]=size(x); %perform a Hilbert transform
m=2^nextpow2(n);
y=fft(real(x),m);
```

```
if m~=1
    h=[1; 2*ones(fix((m-1)/2),1); 1; zeros(fix((m-1)/2),1)];
    y(:)=y.*h(:, ones(1,cc) );
end
```

```
y=ifft(y,m);
y=y(1:n,:);
```

```
if r==1
    y=y.';
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%end of the section based on HILBERT
```

```
envsig=abs(y); %generate the envelope
```

expand.m

```
%function expansig=expand(s,mu,Vm)
%
%EXPAND expands the signal 's' using using the mu-255
%companding law, returning 'expansig'. Values of mu must
%be between 1 and 255; mu = 1 results in a linear function
%(no expansion).
%Input parameters:
% s--the signal for expansion (compressed earlier)
% mu--the degree of expansion effected (mu = 255 delivers the
%   highest expansion, mu = 1 results in no expansion)
% Vm--the maximum voltage in the signal (Note: This value
%   must be exact! Using the 'max' command to pass in the
%   variable ensures its accuracy.)

%Written by Mike Shields 27 Jul 93
%Edited by Susan Guckelberg 5 Dec 93

function expansig=expand(s,mu,Vm);

signs=(2*sign(s))-1; %find the sign of each value of s and adjust to
                    %avoid division by 0
b=s*(log(1+mu)/Vm); %the expansion function

expansig=(Vm/mu)*(exp(b)-1).*signs; %restore the signs to s
```

flattop.m

```
%function [flatsig,pulstrn]=flattop(s,delta_t,samprate,d)
%
%FLATTOP multiplies the signal 's' and an impulse train; then
%convolves it with a pulse to generate the flattop-sampled
%signal 'flatsig'.
%Input parameters:
% s--the signal to be sampled
% delta_t--t2 - t1, the step size of the time vector in the signal
% samprate--the sampling frequency in Hz (note: minimum sampling rate
% is twice the highest frequency in the message signal)
% d--the duty cycle (less than 1)

%Written by Randy Borchardt 27 Jul 93
%Edited by Susan Guckelberg 18 Oct 93

function [flatsig,pulstrn]=flattop(s,delta_t,samprate,d);

length_s=length(s); %number of points in the time vector
tot_time=(length_s-1)*delta_t; %number of seconds in the time vector
T=1/samprate; %sampling period T in seconds

numpulse=tot_time/T; %number of pulses
Tpts=length_s/numpulse; %number of points in sampling period T
taupts=d*Tpts; %number of points in tau

pulse=ones(1,taupts); %generate pulse

imptrn=zeros(1,length_s); %generate impulse train
imp=1:Tpts:length_s;
imptrn(imp)=ones(imp);

pulstrn=conv(imptrn,pulse); %generate pulse train
pulstrn=pulstrn(1,1:length_s);

flatsig1=s.*imptrn; %generate the flattop-sampled signal
flatsig=conv(flatsig1,pulse);
flatsig=flatsig(1,1:length_s); %set length of flatsig to length of s
```


fm_mod.m

```
%function [fm_sig,delta_f,beta]=fm_mod(t,Ac,fc,fm,theta,delta_f,beta)
%
%FM_MOD calculates either delta_f or beta, whichever was not
%passed in, and returns both, along with the frequency-modulated
%signal 'fm_sig'. Generates single- and multi-tone FM signals.
%Input parameters:
% t--the time vector on which the message signal is based
% Ac--the amplitude of the returned frequency modulated signal
% fc--the frequency of the returned frequency modulated signal
% fm--the frequency of the message signal (a vector for multi-tone signals)
% theta--the phase of the message signal (a vector for multi-tone signals)
% delta_f--the maximum frequency deviation (if unknown, pass in 'none', with
% quotes, and it will be calculated and returned)
% beta--the index of modulation (if unknown, pass in 'none', with quotes, and
% it will be calculated and returned)
% Note: Either delta_f or beta MUST be passed in

%Written by Susan Guckelberg 19 Aug 93

function [fm_sig,delta_f,beta]=fm_mod(t,Ac,fc,fm,theta,delta_f,beta);

if beta=='none' %calculate values for beta or delta_f for user
    beta=delta_f/max(fm);
elseif delta_f=='none'
    delta_f=beta*max(fm);
end

sigma=zeros(t); %reserve space for sigma

for i=1:length(fm) %generate the fm signal
    sums(1:length(t))=sin(2*pi*fm(i)*t+theta(i));
    sigma=sigma+sums;
end

fm_sig=Ac*cos(2*pi*fc*t+beta*sigma);
```

freq_div.m

```
%function div_sig=freq_div(squarebspk,t,fc)
%
%FREQ_DIV accepts the squared BPSK signal 'squarebspk' and
%performs a frequency division in preparation for coherent
%detection.
%Input parameters:
% squarebspk--the squared BPSK signal
% t--the time vector for the BPSK signal
% fc--the carrier frequency for the BPSK signal

%Written by Susan Guckelberg 24 Jan 94

function div_sig=freq_div(squarebspk,t,fc);

div_sig=squarebspk./cos(2*pi*fc.*t); %divide the
                                     %squared signal by the carrier
                                     %frequency to shift back to fc
```

fsk.m

```
%function fsksig=fsk(dig_sig,delta_t,bitrate,freq_0,freq_1)
%
%FSK performs frequency shift keying on the input digital
%signal 'dig_sig', returning 'fsksig.'
%Input parameters:
% dig_sig--the digital input signal, usually NRZL
% delta_t--t2 - t1, the step size of the time vector in
% the signal
% freq_0--the frequency assigned to 0-value bits
% freq_1--the frequency assigned to 1-value bits

%Written by Susan Guckelberg 31 Aug 93

function fsksig=fsk(dig_sig,delta_t,bitrate,freq_0,freq_1);

t=(1:length(dig_sig))*delta_t; %regenerate t
T=1/bitrate; %bit duration T in seconds
avg_bitpts=T/delta_t; %number of points in bit

dig_sig=dig_sig(2:length(dig_sig)); %lose the first point

z=1; %index for bits
i=1; %index for dig_sig

while i<length(dig_sig) %generate a frequency vector
    %either high or low
    bitpts=round(z*avg_bitpts)-i+1; %account for fractional
    %values of avg_bitpts
    if dig_sig(i)==0
        freq(i:bitpts+i-1)=ones(1:bitpts)*freq_0; %0 bits
    else
        freq(i:bitpts+i-1)=ones(1:bitpts)*freq_1; %1 bits
    end

    i=i+bitpts;
    z=z+1;
end

freq=[freq(1) freq]; %add one point at beginning
%of vector
fsksig=cos(2*pi*freq.*t);
```

highpass.m

```
%function HPF=highpass(Hz,cutoff)
%
%HIGHPASS generates a simple high-pass filter 'HPF',
%evaluating frequencies in the vector 'Hz'
%and using the cutoff frequency fc.
%Input parameters:
% Hz--a vector representing frequencies in Hz, returned from
%   spectral.m
% fc--the cutoff frequency beginning the filter

function HPF=highpass(Hz,cutoff);

HPF=Hz./(1+(j.*cutoff));
```

hilbert.m

```
%function hilbsig=hilbert(anysig)
%
%HILBERT returns 'hilbsig', the real part of the hilbert transform
%of the input signal 'anysig', showing a 90° phase shift.
%Hilbert is called by the function ssb.m.
%Input parameters:
% anysig--the input signal

%Written by Susan Guckelberg 30 Jul 93

function hilbsig=hilbert(anysig);

length_s=length(anysig);

ffthilb=fft(anysig); %take the fast fourier transform of the signal

                %multiply first half of transform by -j
                %and second half by j
hilbsig=[ffthilb(1) ffthilb(2:ceil(length_s/2)).*(-j) ...
        ffthilb(ceil(length_s/2)+1:length_s).*j];

hilbsig=real(ifft(hilbsig)); %take the inverse fast fourier transform
                        %and return the real part
```

idealbnd.m

```
%function IBPF=idealbnd(Hz,cutoff1,cutoff2)
%
%IDEALBND generates an ideal band-pass filter 'IBPF', evaluating
%frequencies in the vector 'Hz' and using the frequencies
%between 'cutoff1' and 'cutoff2'.
%Input parameters:
% Hz--a vector representing frequencies in Hz, returned from
%   spectral.m
% cutoff1--the frequency beginning the filter
% cutoff2--the frequency ending the filter

%Written by Susan Guckelberg 12 Jul 93

function IBPF=idealbnd(Hz,cutoff1,cutoff2); %2 cutoff frequencies

IBPF=[zeros(1,cutoff1) ones(1,cutoff2-cutoff1) ...
      zeros(1,length(Hz)-cutoff2)];
```

idealhi.m

```
%function IHPF=idealhi(Hz,cutoff)
%
%IDEAHLHI generates an ideal high-pass filter 'IHPF', evaluating
%frequencies in the vector 'Hz' and using the
%frequencies beginning with cutoff.
%Input parameters:
% Hz--a vector representing frequencies in Hz, returned from
%   spectral.m
% cutoff--the frequency beginning the filter

%Written by Susan Guckelberg 12 Jul 93
%Revised 21 Jul 93

function IHPF=idealhi(Hz,cutoff);

ff=cutoff/(Hz(2)-Hz(1));
IHPF=[zeros(1,ff) ones(1,length(Hz)-ff)];
```

ideallow.m

```
%function ILPF=ideallow(Hz,cutoff)
%
%IDEALLOW generates an ideal low-pass filter 'ILPF', evaluating
%frequencies in the vector 'Hz' and using the frequencies
%ending with 'cutoff'.
%Input parameters:
% Hz--a vector representing frequencies in Hz, returned from
%   spectral.m
% cutoff--the frequency ending the filter

%Written by Susan Guckelberg 12 Jul 93
%Revised 10 Aug 93

function ILPF=ideallow(Hz,cutoff);

midfreq=cutoff/(Hz(2)-Hz(1));
ILPF=[ones(1,midfreq) zeros(1,length(Hz)-midfreq)];
```


impsamp.m

```
%function [impsig,imptrn]=impsamp(s,delta_t,samprate)
%
%IMPSAMP multiplies the signal 's' and an impulse train to
%generate the impulse-sampled signal 'impsig'.
%Input parameters:
% s--the signal to be sampled
% delta_t--t2 - t1, the step size of the time vector in the signal
% samprate--the sampling frequency in Hz (note: minimum sampling rate
% is twice the highest frequency in the message signal)

%Written by Randy Borchardt 27 Jul 93
%Edited by Susan Guckelberg 18 Oct 93

function [impsig,imptrn]=impsamp(s,delta_t,samprate);

length_s=length(s); %number of points in the time vector
tot_time=(length_s-1)*delta_t; %number of seconds in the time vector
T=1/samprate; %sampling period T in seconds

numpulse=tot_time/T; %number of pulses
Tpts=length_s/numpulse; %number of points in sampling period T

imptrn=zeros(1,length_s); %generate impulse train
imp=1:Tpts:length_s;
imptrn(imp)=ones(imp);

impsig=s.*imptrn; %generate the impulse-sampled signal
impsig=impsig(1,1:length_s); %set length of impsig to length of s
```

lowpass.m

```
%function LPF=lowpass(Hz,cutoff)
%
%LOWPASS generates a simple low-pass filter 'LPF',
%evaluating frequencies in the vector 'Hz' and using the
%frequencies ending with 'cutoff'.
%Input parameters:
% Hz--a vector representing frequencies in Hz, returned from
% spectral.m
% cutoff--the frequency ending the filter

%Written by Susan Guckelberg 12 Jul 93

function LPF=lowpass(Hz,cutoff);

LPF=1 ./ (1+(j.*(Hz./cutoff)));
```

manchest.m

```
%function [manchsig,bit_t,transec]=manchest(codedsig,delta_t,bitrate)
%
%MANCHEST takes a binary-encoded signal and prepares it for
%plotting as a manchester-coded signal. It returns the
%signal 'manchest', a time vector 'bit_t', and the number of
%seconds needed for transmission 'transec'.
%Input parameters:
% codedsig--a binary-encoded signal
% delta_t--t2 - t1, the step size of the time vector in the signal
% bitrate--number of bits per second desired for the output
% signal
```

```
%Written by Susan Guckelberg 25 Jul 93
```

```
function [manchsig,bit_t,transec]=manchest(codedsig,delta_t,bitrate);
```

```
T=1/bitrate; %sampling period T in seconds
avg_Tpts=(1/delta_t)*T; %number of points in T
```

```
k=1;
i=1;
while i<=length(codedsig) %fill half the sampling period T with
    %either 0's or 1's and the reverse for
    %the remainder
    Tpts=round(i*avg_Tpts)-k+1; %account for fractional
    %values of avg_Tpts
    halfT=round(Tpts/2);
    if codedsig(i)==1
        manchsig(k:k+halfT-1)=ones(1:halfT);
        manchsig(k+halfT+1:k+Tpts)=-1.*ones(1:Tpts-halfT);
    else
        manchsig(k:k+halfT-1)=-1.*ones(1:halfT);
        manchsig(k+halfT+1:k+Tpts)=ones(1:Tpts-halfT);
    end
    k=k+Tpts;
    i=i+1;
end
```

```
transec=length(manchsig)*delta_t; %number of seconds needed for
%transmission of the coded signal
```

```
manchsig=[manchsig(1) manchsig]; %add one point to manchsig
%to account for zero
%in time vector, using value of
%first point
```

```
bit_p=0:delta_t:transec; %time vector for bitstream
```

natsamp.m

```
%function [natsig,pulstrn]=natsamp(s,delta_t,samprate,d)
%
%NATSAMP multiplies the input signal 's' and the pulse train
%'pulstrn' to generate the naturally-sampled signal 'natsig'.
%Input parameters:
% s--the signal to be sampled
% delta_t--t2 - t1, the step size of the time vector in the signal
% samprate--the sampling frequency in Hz (note: minimum sampling rate
% is twice the highest frequency in the message signal)
% d--the duty cycle (less than 1)

%Written by Randy Borchardt 27 Jul 93
%Edited by Susan Guckelberg 18 Oct 93

function [natsig,pulstrn]=natsamp(s,delta_t,samprate,d);

length_s=length(s); %number of points in the time vector
tot_time=(length_s-1)*delta_t; %number of seconds in the time vector
T=1/samprate; %sampling period T in seconds

numpulse=tot_time/T; %number of pulses
Tpts=length_s/numpulse; %number of points in sampling period T
taupts=d*Tpts; %number of points in tau

pulse=ones(1,taupts); %generate pulse

imptrn=zeros(1,length_s); %generate impulse train
imp=1:Tpts:length_s;
imptrn(imp)=ones(imp);

pulstrn=conv(imptrn,pulse); %generate the pulse train
pulstrn=pulstrn(1,1:length_s);

natsig1=s.*pulstrn; %multiply the signal and pulse train
natsig=natsig1(1,1:length_s); %set length of natsig to length of s
```

nrzlbim

```
%function [nrzlsig,bit_t,transec]=nrzlbim(codedsig,delta_t,bitrate)
%
%NRZLBI takes a binary-encoded signal and prepares it for plotting
%as a non-return-to-zero bipolar level signal. It returns the
%signal 'nrzlsig', the time vector 'bit_t', and the number of
%seconds needed for transmission 'transec'.
%Input parameters:
% codedsig--a binary-encoded signal
% delta_t--t2 - t1, the step size of the time vector in the signal
% bitrate--number of bits per second desired for the output
% signal

%Algorithm by Pete Hutson 17 Mar 93
%Written by Susan Guckelberg 25 Jul 93

function [nrzlsig,bit_t,transec]=nrzlbim(codedsig,delta_t,bitrate);

T=1/bitrate; %bit duration T in seconds
avg_Tpts=(1/delta_t)*T; %number of points in T

k=1; %index for nrzlsig
i=1; %index for codedsig

while i<=length(codedsig) %fill T with
    %either 0's or 1's
    Tpts=round(i*avg_Tpts)-k+1; %account for fractional
    %values of avg_Tpts
    if codedsig(i)==1
        nrzlsig(k:k+Tpts-1)=ones(1:Tpts);
    else
        nrzlsig(k:k+Tpts-1)=ones(1:Tpts)*(-1);
    end
    k=k+Tpts;
    i=i+1;
end

transec=length(nrzlsig)*delta_t; %number of seconds needed for
    %transmission of the coded signal

nrzlsig=[nrzlsig(1) nrzlsig]; %add one point to nrzlsig
    %to account for zero
    %in time vector, using value of
    %first point

bit_t=0:delta_t:transec; %time vector for bitstream
```

nrzluni.m

```
%function [nrzlsig,bit_t,transec]=nrzluni(codedsig,delta_t,bitrate)
%
%NRZLUNI takes a binary-encoded signal and prepares it for plotting
%as a non-return-to-zero unipolar level signal. It returns the
%signal 'nrzlsig', the time vector 'bit_t', and the number of
%seconds needed for transmission 'transec'.
%Input parameters:
% codedsig--a binary-encoded signal
% delta_t--t2 - t1, the step size of the time vector in the signal
% bitrate--number of bits per second desired for the output
% signal

%Algorithm by Pete Hutson 17 Mar 93
%Written by Susan Guckelberg 25 Jul 93

function [nrzlsig,bit_t,transec]=nrzluni(codedsig,delta_t,bitrate);

T=1/bitrate; %bit duration T in seconds
avg_Tpts=(1/delta_t)*T; %number of points in T

k=1; %index for points in nrzlsig
i=1; %index for points in codedsig
while i<=length(codedsig) %fill T with
    %either 0's or 1's
    Tpts=round(i*avg_Tpts)-k+1; %account for fractional
    %values of avg_Tpts
    if codedsig(i)==1
        nrzlsig(k:k+Tpts-1)=ones(1:Tpts);
    else
        nrzlsig(k:k+Tpts-1)=zeros(1:Tpts);
    end
    k=k+Tpts;
    i=i+1;
end

transec=length(nrzlsig)*delta_t; %number of seconds needed for
    %transmission of the coded signal
nrzlsig=[nrzlsig(1) nrzlsig]; %add one point to nrzlsig
    %to account for zero
    %in time vector, using value of
    %first point

bit_t=0:delta_t:transec; %time vector for bitstream
```

par_ser.m

```
%function comb_sig=par_ser(odd_sig,even_sig,delta_t,bitrate)
%
%PAR_SER accepts the odd and even signals previously separated in
%the function 'ser_par.m' and combines them at twice their
%bitrate, returning 'comb_sig'.
%Input parameters:
% odd_sig--the digital signal composed of odd bits, returned
%   from 'ser_par.m'
% even_sig--the digital signal composed of even bits, returned
%   from 'ser_par.m'
% delta_t--t2 - t1, the step size of the time vector in the signal
% bitrate--number of bits per second in the original digital
%   signal (before separation)

function comb_sig=par_ser(odd_sig,even_sig,delta_t,bitrate);

odd_sig=odd_sig(2:length(odd_sig)); %remove last value, added
even_sig=even_sig(2:length(even_sig)); %to account for 0 in time vector

bit_dur=1/bitrate; %bit duration in seconds
avg_bitpts=(1/delta_t)*bit_dur; %number of points in each bit

k=1; %index for bits
i=1; %index for length of vector
      %generate combined signal at
while i<length(odd_sig) %twice the bitrate passed in

    bitpts=round(k*avg_bitpts)-i+1; %account for fractional
      %values of avg_bitpts
    comb_sig(i:i+bitpts-1)=odd_sig(i:i+bitpts-1);
    i=i+bitpts;
    k=k+1;
    bitpts=round(k*avg_bitpts)-i+1; %account for fractional
      %values of avg_bitpts
    comb_sig(i:i+bitpts-1)=even_sig(i:i+bitpts-1);
    k=k+1;
    i=i+bitpts;

end

comb_sig=[comb_sig(1) comb_sig]; %add one point to comb_sig
      %to account for zero
      %in time vector, using value of
      %first point
```


psd.m

```
%function [psdsig,Hz,fftsig]=psd(anysig,delta_t)
%
%PSD performs a fast fourier transform on the input signal 'anysig'
%and returns 'fftsig'. For a power spectral density graph,
%plot 'psdsig' against 'Hz'. To find average signal power,
%integrate psdsig by using the 'sum' command.
%Input parameters:
% anysig--the input signal
% delta_t--t2 - t1, the step size of the time vector in the signal

%Written by Susan Guckelberg, 5 Aug 93

function [psdsig,Hz,fftsig]=psd(anysig,delta_t);

fftsig=fft(anysig); %find the fast Fourier transform of the signal

abfftsig=abs(fftsig); %find the absolute value of the transform

length_f=length(abfftsig); %find the length of the fourier transform

shortsig=[abfftsig(1) abfftsig(2:ceil((length_f)/2))]; %excluding first
                                                    %value, truncate the vector
                                                    % to half its length

lengthss=length(shortsig); %find length of transform vector

psdsig=shortsig./lengthss; %scale down amplitude by
                            %dividing the frequency values
                            %by the vector length

psdsig=(psdsig.^2)/2; %square and halve the amplitudes

nyqfreq=1/(delta_t*2); %find the Nyquist frequency

Hz=nyqfreq*(1:lengthss)/lengthss; %create Hz frequency vector
```

pulspos.m

```
%function ppsig=pulspos(s,delta_t,samprate,pulsdur)
%
%PULSPOS returns the pulse-position-modulated signal 'ppsig' for
%the signal 's'. Pulse amplitude is set at half of the maximum
%amplitude of the signal. Offset values are calculated from the
%beginning of the sampling period T.
%Input parameters:
% s--the signal to be sampled
% delta_t--t2 - t1, the step size of the time vector in the signal
% samprate--the sampling frequency in Hz (note: minimum sampling rate
% is twice the highest frequency in the carrier)
% pulsdur--the pulse duration (fraction of the sampling period T). Maximum
% pulse offset is T - pulsdur.
% (Note: pulsedur is typically small; the larger the pulse duration,
% the smaller the maximum pulse offset)

%Written by Susan Guckelberg, 6 Jul 93
%Revised 8 Nov 93

function ppsig=pulspos(s,delta_t,samprate,pulsdur);

length_s=length(s); %number of points in the time vector

tot_time=length_s*delta_t; %number of seconds in the time vector
T=1/samprate; %sampling period T in seconds

numpulse=tot_time/T; %number of pulses
avg_Tpts=length_s/numpulse; %avg number of points in sampling period T
taupts=ceil(avg_Tpts*pulsdur); %set the pulse duration tau,
%avoiding zero values
minamp=min(s); %find the minimum amplitude in s
maxamp=max(s); %find the maximum amplitude in s
pulsamp=maxamp/2; %set the pulse amplitude
pos_s=s+abs(minamp)+.01; %increase s values to all positive
max_pos=max(pos_s); %find the maximum amplitude in pos_s

i=1; %initialize index
pulsenum=1;
while i<=length_s %generate pulse position modulated signal
    pulsperc=pos_s(i)/max_pos; %find the percentage of each pulse
    %amplitude to the maximum amplitude
    Tpts=round(pulsenum*avg_Tpts)-i+1; %account for fractional
    %values of avg_Tpts
    mxoffpts=Tpts-taupts; %find maximum offset value

    prezero=ceil(pulsperc*mxoffpts); %assign the amplitude percentage
```

```

                                %to the pulse offset
postzero=Tpts-taups-prezero; %find the number of points
                                %needed to fill T
if Tpts==taups
    pp=pulsamp*ones(1:Tpts); %pulse extends the entire width of T
else
    %zero level signal points preceding pulse
    pulse(1:prezero)=pulsperc*zeros(1:prezero);
    %pulse points with value pulsamp
    pulse(prezero+1:prezero+taups)=pulsamp*ones(1:taups);
    %zero values for remainder of T
    pulse(prezero+taups+1:Tpts)=zeros(1:postzero);
end
    %generate the pulse position modulated signal ppsig
    ppsig(i:i+Tpts-1)=pulse(1:Tpts);
    i=i+Tpts;
    pulsenum=pulsenum+1;
end

if length(ppsig)<length_s %add zeros to ppsig if shorter than s
    ppsig=[ppsig zeros(1:length_s-length(ppsig))];
end

if length(ppsig)>length_s %truncate ppsig if longer than s
    ppsig=ppsig(1:length_s);
end

```

pulswid.m

```
%function pwsig=pulswid(s,delta_t,samprate,maxdur)
%
%PULSWID returns the pulse-width-modulated signal 'pwsig' for the
%signal 's'. The amplitude of pwsig is half the maximum amplitude
%of s.
%Input parameters:
% s--the signal to be modulated
% delta_t--t2 - t1, the step size of the time vector in the signal
% samprate--the sampling frequency in Hz (note: minimum sampling rate
% is twice the highest frequency in the carrier)
% maxdur--the maximum pulse duration, expressed as a fraction of
% the sampling period T (Note: maximum can be 1, typically close to 1)

%Written by Susan Guckelberg, 6 Jul 93
%Revised 7 Nov 93

function pwsig=pulswid(s,delta_t,samprate,maxdur);

length_s=length(s); %number of points in the time vector

tot_time=length_s*delta_t; %number of seconds in the time vector
T=1/samprate; %sampling period T in seconds
numpulse=tot_time/T; %number of pulses
avg_Tpts=length_s/numpulse; %avg number of points in sampling period T

minamp=min(s); %find the minimum amplitude in s
maxamp=max(s); %find the maximum amplitude in s
pulsamp=maxamp/2; %set the pulse amplitude
pos_s=s+abs(minamp)+.01; %increase s values to all positive
max_pos=max(pos_s); %find the maximum amplitude in pos_s

i=1; %initialize index
pulsenum=1;
while i<=length_s %generate pulse width modulated signal
    pulspcr=pos_s(i)/max_pos; %find the percentage of the
    %amplitude to the maximum amplitude

    Tpts=round(pulsenum*avg_Tpts)-i+1; %account for fractional
    %values of avg_Tpts
    maxpuls=Tpts*maxdur; %set the maximum pulse duration

    taupls=ceil(pulspcr*maxpuls); %assign the percentage to the maximum
    %pulse duration to find the number
    %of points in tau, avoiding zero values

    %generate the pulse width modulated signal
```

```

pwpulse(1:taupts)=pulsamp*ones(1:taupts);    %pulse points with
                                              %value pulsamp
pwpulse(taupts+1:Tpts)=zeros(1:Tpts-taupts); %zero values for
                                              %remainder of T
pwsig(i:i+Tpts-1)=pwpulse(1:Tpts);
i=i+Tpts;
pulsenum=pulsenum+1;
end

if length(pwsig)<length_s    %add zeros to pwsig if shorter than s
    pwsig=[pwsig zeros(1:length_s-length(pwsig))];
end

if length(pwsig)>length_s    %truncate pwsig if longer than s
    pwsig=pwsig(1:length_s);
end

```

quantize.m

```
%function [quanch_x,quanch_y,quan_sig,bin_nums]=quantize(symbols,
%elements,ss,samprate,delta_t)
%
%QUANTIZE accepts a signal 'ss' ranging between -10 and 10 volts.
%It returns 'bin_nums', a vector containing the bin numbers
%for the quantization scheme (bin numbers begin with 0);
%the bipolar quantized signal 'quan_sig' (values are rounded
%rather than truncated); and vectors 'quanch_x' and 'quanch_y',
%for use in plotting the quantization characteristic.
%For quantization system characteristic only, input only 'symbols'
%and 'elements', and define outputs 'quanch_x' and 'quanch_y'.
%Input parameters:
% ss--the sampled signal to be quantized (usually a flattop-
%   sampled signal)
% samprate--the sampling frequency in Hz
% delta_t--t2 - t1, the step size of the time vector in the signal
% symbols--the number of symbols in the coding scheme
%   (binary=2, quad=4, hex=16)
% elements--the number of places in the coding scheme
%   (the power to which the number of symbols is raised)

%Written by Susan Guckelberg, 21 Jul 93
%Revised 23 Nov 93

function[quanch_x,quanch_y,quan_sig,bin_nums]=quantize(symbols,...
elements,ss,samprate,delta_t);

maxqs=10; %maximum value in the quantized signal
minqs=-10; %minimum value in the quantized signal

levels=symbols^elements; %number of levels in the
                        %quantization scheme
bins=levels-1;
stair=(maxqs-minqs)/levels; %size of the vertical step in
                        %the quantization scheme
for k=1:levels+1 %set the quantization characteristics
    quanch_y(k)=minqs+((k-1)*stair);
    if k==1
        quanch_x(k)=minqs;
    elseif k==levels+1
        quanch_x(k)=maxqs;
        quanch_y(k)=minqs+((k-2)*stair);
    else quanch_x(k)=minqs+((k-1.5)*stair);
    end
end
end
```

```

if nargin==5 %use when a signal is input for quantization

lengthss=length(ss);
tot_time=(lengthss-1)*delta_t; %number of seconds in the time vector
T=1/samprate; %sampling period T in seconds

numpulse=tot_time*samprate; %number of pulses
avg_Tpts=lengthss/numpulse; %avg number of points in sampling period T

i=1; %initialize index i, q, and pulsenum
q=1;
pulsenum=1;
while i<lengthss
    level=minqs; %initialize minimum quantized signal level
    j=1; %initialize index
    while ss(i)>quanch_y(j)+.5*stair %check for values halfway
        level=quanch_y(j+1); %between steps
        j=j+1;
        if j==levels %maximum level reached
            break
        end
    end
    Tpts=round(pulsenum*avg_Tpts)-i+1; %account for fractional
        %values of avg_Tpts
    quan_sig(i:Tpts+i-1)=level*ones(1:Tpts); %generate quantized signal
    bin_nums(q)=j-1; %bin numbers begin with 0
    q=q+1;
    i=i+Tpts;
    pulsenum=pulsenum+1;
end

if length(quan_sig)<lengthss %pad quan_sig with last assigned value if
    %shorter than ss
    quan_sig=[quan_sig ss(i)*ones(1:lengthss-length(quan_sig))];
end

if length(quan_sig)>lengthss %truncate quan_sig if longer than ss
    quan_sig=quan_sig(1:lengthss);
end

end %end if

```

quantuni.m

```
%function [quanch_x,quanch_y,quan_sig,bin_nums]=quantuni(symbols,  
%elements,ss,samprate,delta_t)  
%  
%QUANTUNI accepts a signal 'ss' ranging between 0 and 10 volts.  
%It returns 'bin_nums', a vector containing the bin numbers  
%for the quantization scheme (bin numbers begin with 0);  
%the bipolar quantized signal 'quan_sig' (values are truncated  
%rather than rounded); and vectors 'quanch_x' and 'quanch_y',  
%for use in plotting the quantization characteristic.  
%For quantization system characteristic only, input only 'symbols'  
%and 'elements', and define outputs 'quanch_x' and 'quanch_y'.  
%Input parameters:  
% ss--the sampled signal to be quantized (usually a flattop-  
%   sampled signal)  
% samprate--the sampling frequency in Hz  
% delta_t--t2 - t1, the step size of the time vector in the signal  
% symbols--the number of symbols in the coding scheme  
%   (binary=2, quad=4, hex=16)  
% elements--the number of places in the coding scheme  
%   (the power to which the number of symbols is raised)
```

```
%Written by Susan Guckelberg, 21 Jul 93
```

```
%Revised 23 Nov 93
```

```
function[quanch_x,quanch_y,quan_sig,bin_nums]=quantuni(symbols,...  
elements,ss,samprate,delta_t);
```

```
maxqs=10; %maximum value in the quantized signal  
minqs=0; %minimum value in the quantized signal
```

```
levels=symbols^elements; %number of levels in the  
                        %quantization scheme
```

```
bins=levels-1;
```

```
stair=(maxqs-minqs)/levels; %size of the vertical step in  
                        %the quantization scheme
```

```
for k=1:levels+1 %set the quantization characteristics
```

```
    quanch_y(k)=minqs+((k-1)*stair);
```

```
    if k==1
```

```
        quanch_x(k)=minqs;
```

```
    elseif k==levels+1
```

```
        quanch_x(k)=maxqs;
```

```
        quanch_y(k)=minqs+((k-2)*stair);
```

```
    else quanch_x(k)=minqs+((k-1)*stair);
```

```
    end
```

```
end
```



```

if nargin==5 %use when a signal is input for quantization

lengthss=length(ss);
tot_time=(lengthss-1)*delta_t; %number of seconds in the time vector
T=1/samprate; %sampling period T in seconds

numpulse=tot_time*samprate; %number of pulses
avg_Tpts=lengthss/numpulse; %avg number of points in sampling period T

i=1; %initialize index i, q, and pulsenum
q=1;
pulsenum=1;
while i<lengthss
    level=max(quanch_y); %initialize maximum quantized signal level
    j=levels; %initialize index
    while ss(i)<quanch_y(j)
        level=quanch_y(j-1);
        j=j-1;
        if j==0 %minimum level reached
            break
        end
    end
    Tpts=round(pulsenum*avg_Tpts)-i+1; %account for fractional
        %values of avg_Tpts
    quan_sig(i:Tpts+i-1)=level*ones(1:Tpts); %generate quantized signal
    bin_nums(q)=j-1; %bin numbers begin with 0
    q=q+1;
    i=i+Tpts;
    pulsenum=pulsenum+1;
end

if length(quan_sig)<lengthss %pad quan_sig with last assigned value if
    %shorter than ss
    quan_sig=[quan_sig ss(i)*ones(1:lengthss-length(quan_sig))];
end

if length(quan_sig)>lengthss %truncate quan_sig if longer than ss
    quan_sig=quan_sig(1:lengthss);
end

end %end if

```

recoverm.m

```
%function recsig=recoverm(fftsig,func,H,cutoff1,cutoff2)
%
%RECOVERM returns 'recsig', a filtered and recovered modulated
%signal.
%Input parameters:
% fftsig--the transform to be filtered and recovered
% func--the title of the filter function, which must be passed
%   in as a string enclosed in quotes
%   Filter functions in the Communications Toolbox are
%   LOWPASS,HIGHPASS,IDEALLOW,IDEALHI, and IDEALBND.
% H--the vector representing frequency in Hz, returned from spectral.m
% cutoff1--the cutoff frequency for low- and high-pass filters
% cutoff2--an additional cutoff frequency, passed in only for bandpass filters

%Written by Susan Guckelberg 10 Aug 93

function recsig=recoverm(fftsig,func,H,cutoff1,cutoff2);

if nargin==4
    halffltr=feval(func,H,cutoff1); %generate the filter vector
elseif nargin==5
    halffltr=feval(func,H,cutoff1,cutoff2); %use fc2 for bandpass filter
end
    %complete the filter
fltr=[halffltr(1:length(halffltr)-1) fliplr(halffltr)];

fltsig=fltr.*fftsig; %multiply the filter and signal

recsig=real(ifft(fltsig)); %perform inverse fast fourier transform
```

recovers.m

```
%function recsig=recovers(fftsig,d,func,H,cutoff1,cutoff2)
%
%RECOVERS returns 'recsig', a filtered and recovered sampled signal.
%Input parameters:
% fftsig--the transform vector to be filtered and recovered, returned
%   from spectral.m
% d--the duty cycle of the sampled signal. (Note: to recover an impulse-
%   sampled signal, calculate d=samprate*delta_t.)
% func--the title of the filter function, passed
%   in as a string enclosed in single quotes
%   Filter functions in the Communications Toolbox are
%   LOWPASS,HIGHPASS,IDEALLOW,IDEALHI, and IDEALBND.
% H--the vector representing frequency in Hz, returned from spectral.m
% cutoff1--the cutoff frequency for low- and high-pass filters
% cutoff2--an additional cutoff frequency, passed in only for bandpass filters

%Written by Susan Guckelberg, 7 Jul 93
%Revised 10 Aug 93

function recsig=recovers(fftsig,d,func,H,cutoff1,cutoff2);

if nargin==5
    halffilt=feval(func,H,cutoff1); %generate the filter vector
elseif nargin==6
    halffilt=feval(func,H,cutoff1,cutoff2); %for bandpass filters
end

    %complete the filter
    filt=[halffilt(1:length(halffilt)-1) fliplr(halffilt)];

    filtsig=filt.*fftsig.*(1/d); %multiply the filter and signal
    %use 1/d to scale the signal amplitude

    recsig=ifft(filtsig); %perform inverse fast fourier transform
```

rzuni.m

```
%function [rzunisig,bit_t,transec]=rzuni(codedsig,delta_t,bitrate)
%
%RZUNI takes a binary-encoded signal and prepares it for plotting
%as a return-to-zero unipolar coded signal. It returns the signal
%'rzunisig' and the number of seconds needed for transmission
%'transec'.
%Input parameters:
% codedsig--a binary-encoded signal
% delta_t--t2 - t1, the step size of the time vector in the signal
% bitrate--number of bits per second desired for the output
% signal

%Written by Susan Guckelberg 25 Jul 93

function [rzunisig,bit_t,transec]=rzuni(codedsig,delta_t,bitrate);

T=1/bitrate; %bit duration T in seconds
avg_Tpts=(1/delta_t)*T; %number of points in T

k=1; %index for rzunisig
i=1; %index for codedsig
while i<=length(codedsig) %fill half the sampling period T with
    %either 0's or 1's and zeros for
    %the remainder
    Tpts=round(i*avg_Tpts)-k+1; %account for fractional
    %values of avg_Tpts
    halfT=round(Tpts/2);
    if codedsig(i)==1
        rzunisig(k:k+halfT-1)=ones(1:halfT);
    else
        rzunisig(k:k+halfT-1)=zeros(1:halfT);
    end
    rzunisig(k+halfT+1:k+Tpts)=zeros(1:Tpts-halfT);
    k=k+Tpts;
    i=i+1;
end

transec=length(rzunisig)*delta_t; %number of seconds needed for
    %transmission of the coded signal

rzunisig=[rzunisig(1) rzunisig]; %add one point to rzunisig
    %to account for zero
    %in time vector, using value of
    %first point

bit_t=0:delta_t:transec; %time vector for bitstream
```

ser_par.m

```
%function [odd_sig,even_sig]=ser_par(dig_sig,delta_t,bitrate)
%
%SER_PAR accepts a digital signal (one that is a function of a
%time vector, such as those returned from 'rzuni.m' or
%'nrzlb.m') and separates it into two digital signals, one
%composed of the odd bits and one of the even bits. The bit
%durations in the output digital signals are twice that of the
%input digital signal. Input signals ARE RESTRICTED to those
%with EVEN numbers of bits.
%Input parameters:
% dig_sig--the digital signal, a function of a time vector
% delta_t--t2 - t1, the step size of the time vector in the signal
% bitrate--number of bits per second in the input signal
%   dig_sig

%Written by Susan Guckelberg 6 Sep 93

function [odd_sig,even_sig]=ser_par(dig_sig,delta_t,bitrate);

dig_sig=dig_sig(2:length(dig_sig)); %remove first value, added
                                     %to account for 0 in time vector
tot_time=length(dig_sig)*delta_t; %number of seconds in the time vector
bit_dur=1/bitrate; %bit duration in seconds
numbits=round(tot_time*bitrate); %number of bits in input signal

avg_bitpts=(1/delta_t)*bit_dur; %number of points in each bit

odd_sig=[];
even_sig=[];
z=1; %index for counting bits
i=1; %index for dig_sig

    %generate odd and even bitstreams at half the bitrate

while i<length(dig_sig)
    bitpts=round(z*avg_bitpts)-i+1; %find length of the odd
                                     %bit in dig_sig, accounting
                                     %for fractional values
                                     %of avg_bitpts

    bit(1:bitpts)=ones(1:bitpts)*dig_sig(i); %set odd bit to the
                                               %value of dig_sig(i)
    odd_sig=[odd_sig bit]; %add bit to odd_sig
    bit=[]; %important to wipe out the bit
    i=i+bitpts; %advance the index in dig_sig, keep bitpts the same
    bit(1:bitpts)=ones(1:bitpts)*dig_sig(i); %assign value of
```

```

even_sig=[even_sig bit];          %next bit in dig_sig
bit=[];                            %to even_sig
z=z+1;    %advance bit number in dig_sig
bitpts=round(z*avg_bitpts)-i+1; %find length of next bit
                                %in dig_sig

bit(1:bitpts)=ones(1:bitpts)*odd_sig(length(odd_sig));
odd_sig=[odd_sig bit]; %add the bit to odd_sig
bit=[];
bit(1:bitpts)=ones(1:bitpts)*even_sig(length(even_sig));
even_sig=[even_sig bit]; %add the bit to even_sig
bit=[];
i=i+bitpts; %advance index in dig_sig
z=z+1; %advance one bit in dig_sig
end

odd_sig=[odd_sig(1) odd_sig]; %add one point to odd signal
                                %to account for zero
                                %in time vector, using value of
                                %end point
even_sig=[even_sig(1) even_sig]; %repeat for even signal

```

snr.m

```
%function s_n_ratio=snr(cleansig,noisysig)
%
%SNR subtracts a noisy signal (usually the output or
%recovered signal with noise) from a clean signal
%(usually the input or message signal) and
%returns the signal to noise ratio 's_n_ratio' in dB
%Input parameters:
% cleansig--signal vector
% noisysig--signal vector

%Written by Mike Shields 28 Jul 93
%Edited by Susan Guckelberg 5 Dec 93

function s_n_ratio=snr(cleansig,noisysig);

noise=cleansig-noisysig;

s_n_ratio=10*log10(sum(cleansig.^2)/sum(noise.^2)); %snr in dB
```

spectral.m

```
%function [specsig,HZ,fftsig]=spectral(anysig,delta_t)
%
%SPECTRAL performs a fast fourier transform on the input signal 'anysig'
%and returns 'fftsig'. For a spectral analysis graph, plot 'specsig'
%against 'Hz'.
%Input parameters:
% anysig--the input signal
% delta_t--t2 - t1, the step size of the time vector in the signal

%Written by Susan Guckelberg, 6 Jul 93

function [specsig,HZ,fftsig]=spectral(anysig,delta_t);

fftsig=fft(anysig); %find the fast fourier transform of the signal

abfftsig=abs(fftsig); %find the absolute value of the transform

length_f=length(abfftsig); %find the length of the fourier transform

shortsig=[abfftsig(1) abfftsig(2:ceil((length_f)/2))]; %excluding first
                                                    %value, truncate the
                                                    %vector to half its length

lengthss=length(shortsig); %find length of transform vector

specsig=shortsig./lengthss; %scale down amplitude by
                            %dividing the frequency values
                            %by the vector length

nyqfreq=1/(delta_t*2); %find the Nyquist frequency

Hz=nyqfreq*(1:lengthss)/lengthss; %create Hz frequency vector
```


ssb.m

```
%function [lsbsig,usbsig]=ssb(t,msg,car_amp,car_freq)
%
%SSB calls hilbert.m to perform a hilbert transform
%on the input signal, returning lower and upper sideband
%signals 'lsbsig' and 'usbsig'.
%Input parameters:
% t--the time vector used to generate the message signal
% msg--the message signal
% car_amp--the amplitude of the carrier (modulating) signal
% car_freq--the frequency of the carrier (modulating) signal

%Written by Susan Guckelberg 31 Jul 93

function [lsbsig,usbsig]=ssb(t,msg,car_amp,car_freq);

m_hat=hilbert(msg);

ssb1=msg.*cos((car_freq*2*pi).*t);
ssb2=m_hat.*sin((car_freq*2*pi).*t);

clear m_hat

lsbsig=(ssb1+ssb2).*(car_amp/2);
usbsig=(ssb1-ssb2).*(car_amp/2);
```

**APPENDIX F—USER'S GUIDE TO THE
COMMUNICATIONS TOOLBOX FOR MATLAB**



User's Guide to the

***Communications Toolbox
for MATLAB***

Susan Guckelberg



Table of Contents

purpose	5
platforms	5
references	5
amplitude modulation	
conv_am, hilbert, ssb.....	9
detection	
envelope.....	10
digital encoding	
convert, encode.....	8
manchest, nrzlb, nrzluni, rzuni.....	16
filters	
highpass, idealbnd, idealhi, ideallow, lowpass.....	15
frequency modulation	
fm_mod.....	12
pulse modulation	
pulpos, pulswid.....	19
quantization	
compress, expand.....	7
quantize, quantuni.....	20
snr.....	22
radio frequency digital modulation	
freq_div.....	13
fsk.....	14
par_ser, ser_par.....	17
sampling	
flattop, impsamp, natsamp.....	11
signal recovery	
recoverm, recovers.....	21
spectral analysis	
psd, spectral.....	18
index	23

purpose

These thirty-four functions support basic electronic communication concepts and techniques, and were designed to accompany the EO 3513 course offered at Naval Postgraduate School, Monterey, California.

The "help" function in MATLAB provides quick access to detailed information about required input parameters.

platforms

The Communications Toolbox for use with MATLAB was developed on a Macintosh Powerbook 165c using MATLAB SIMULINK version 1.2, by The MathWorks, Inc. The MATLAB m-files were translated for use with MS-DOS platforms using MacLink Plus 6.0, by DataVis, Inc.

The Communications Toolbox is designed to run under MATLAB 3.5 for the IBM PC, without the presence of other specialized toolboxes.

MATLAB is a registered trademark of The MathWorks, Inc.

references

Brown, LT Dennis W. and Fargues, Monique P., Department of Electrical and Computer Engineering, Naval Postgraduate School Technical Report no. NPSEC-93-017, *SPC Toolbox*, 15 October 1993.

Couch, Leon W., II, *Digital and Analog Communication Systems*, Macmillan Publishing Company, 1993.

Haykin, Simon, *Communication Systems*, Second Edition, John Wiley & Sons Inc., 1983.

The MathWorks, Inc., *The Student Edition of MATLAB for Macintosh Computers*, Prentice-Hall, Inc.

Schweber, William, *Electronic Communications Systems, A Complete Course*, Prentice-Hall, Inc., 1991.

Stanley, William D., *Electronic Communications Systems*, Prentice-Hall, Inc., 1982.

User's Guide to the Communications Toolbox for MATLAB—page 5

compress, expand

function calls:

```
comsig=compress(s,mu,Vm)  
expansig=expand(comsig,mu,Vm)
```

synopsis:

compress and **expand**, used together, simulate the companding process for reducing quantization noise (Schweber, p. 342). The value of V_m (the maximum amplitude, or voltage) in the signal "s" can be identified at the time of the function call by using the "max" command to pass in the "Vm" parameter, as demonstrated with **expand**:

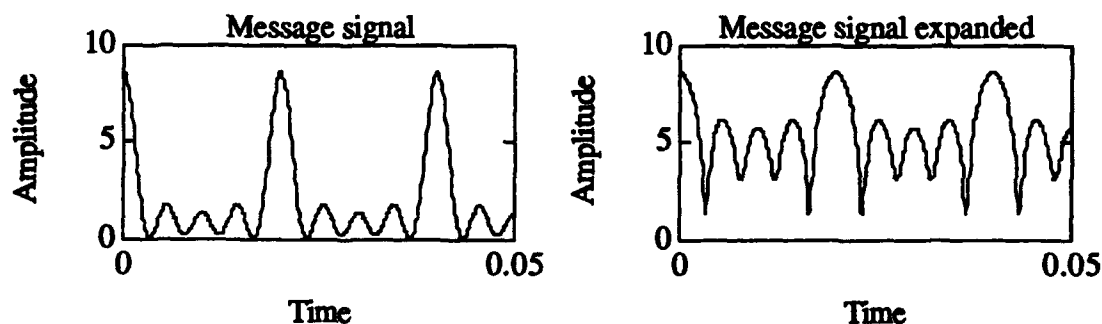
```
expansig=expand(comsig,255,max(comsig));
```

algorithm:

The μ -255 companding law is described by the formula:

$$V_{out} = \frac{V_{max} \left[\ln \left(1 + \frac{\mu V_{in}}{V_{max}} \right) \right]}{\ln(1 + \mu)}$$

where $1 \leq \mu \leq 255$. $\mu = 0$ results in a linear function (no compression or expansion).



convert, encode

function calls:

```
conv_num=convert(decimal,symbols,elements)
codedsig=encode(bin_nums,symbols,elements)
```

synopsis:

Convert and **encode** are used to transform the vector of bin numbers returned from **quantize** or **quantuni** to a stream of values encoded in other than decimal representation, commonly as a bitstream of 1's and 0's. **Encode** calls **convert** for each value in the "bin_nums" vector.

algorithm:

Encode finds the length of the vector "bin_nums" and loops through the vector, passing each value to **convert** along with the number of symbols (base) and elements (places) desired for the output.

In **convert**, the MATLAB function "rem" is used to repeatedly evaluate the input base 10 number "decimal" resulting in a new number of base "symbol" having the number of places specified by "elements." For example, input of 2 symbols and 3 elements results in a three-bit binary number. Each value returned from **convert** is added to the vector "codedsig."

conv_am, hilbert, ssb

function calls:

```
convansig=conv_am(msg,delta_t,fc,m)
hilbsig=hilbert(anysig)
[lsbsig,usbsig]=ssb(t,msg,car_amp,car_freq)
```

synopsis:

conv_am and **ssb** perform two types of amplitude modulation (AM): conventional AM, and single sideband AM. (The third type, double sideband AM, can be produced by multiplying the message signal by a cosine function of the desired carrier frequency.)

algorithm:

conv_am returns a conventional AM signal by employing the formula

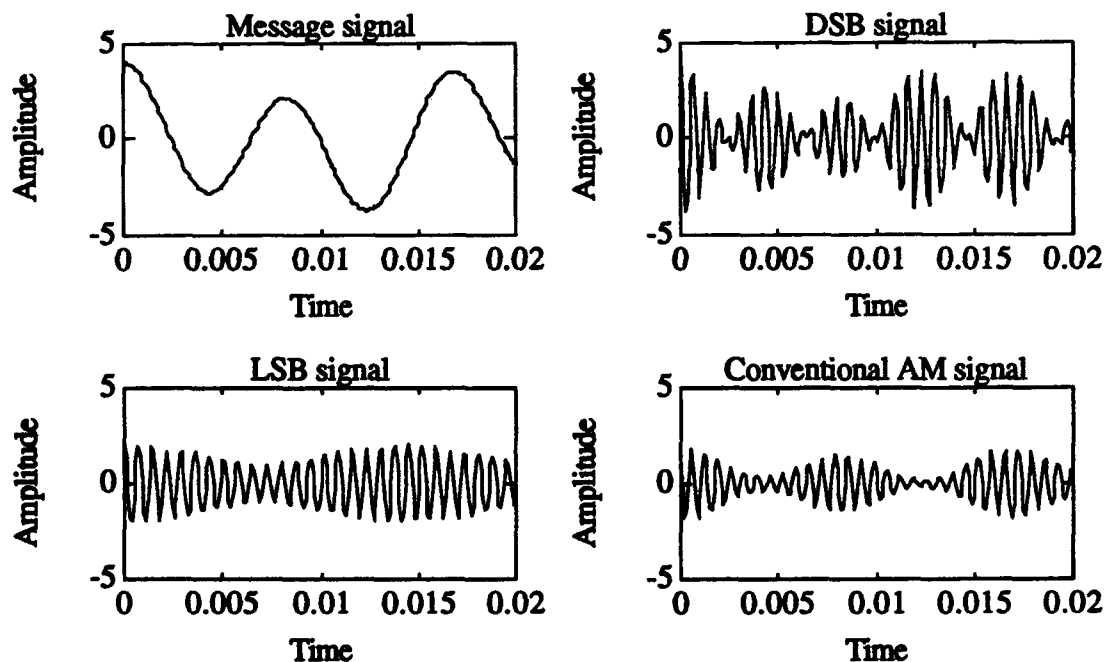
$$A[1 + m_x(t)]\cos \omega_c t$$

where $x(t)$ is the message signal and m is the modulation index (Stanley, p. 147).

ssb returns the lower and upper single sideband (SSB) signals as follows:

$$A_c[m(t)\cos \omega_c t \pm \hat{m}(t)\sin \omega_c t]$$

where $m(t)$ is the message signal. The minus (-) sign is used to produce the upper sideband (USB) signal; the plus (+) sign is used to produce the lower sideband (LSB) signal. The value of $\hat{m}(t)$ is determined within the function **hilbert** by taking the fast fourier transform of the message signal and applying -j to the positive frequencies and j to the negative frequencies (Couch, p. 314).



envelope

function call:

`envsig=envelope(x)`

synopsis:

envelope performs an envelope detection on the input signal "x." (Note: **envelope** performs a transpose of the input signal vector, making it a memory-intensive function. Reduce the size of the input vector whenever possible.)

Written by Dennis W. Brown 8 Sep 93

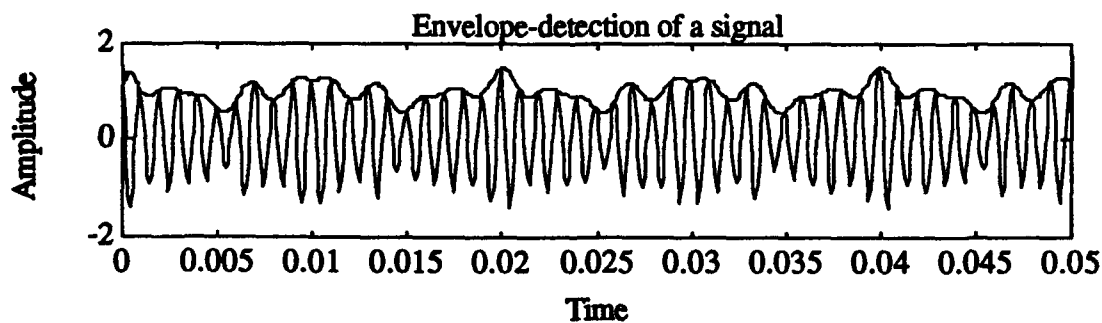
Hilbert transform section written by Charles Denham 7 Jan 88

Revised by LS, 19 Nov 88, 22 May 90, TPK 4 Nov 92

Copyrighted by The MathWorks, Inc. 1988,1990,1992 (The MathWorks, Inc., p. 310)

algorithm:

envelope computes the Hilbert transform of "x," resulting in the magnitude of the complex envelope (Brown and Fargues, p. 55).



User's Guide to the Communications Toolbox for MATLAB—page 10

flattop, impsamp, natsamp

function calls:

```
[flatsig,pulstrn]=flattop(s,delta_t,samprate,d)
[impsig,imptrn]=impsamp(s,delta_t,samprate)
[natsig,pulstrn]=natsamp(s,delta_t,samprate,d)
```

synopsis:

flattop, **impsamp**, and **natsamp** sample the input signal "s." **flattop** returns a flattop-sampled signal; **impsamp** returns an impulse-sampled signal; **natsamp** returns a naturally-sampled signal.

Written by Randy Borchardt 27 Jul 93

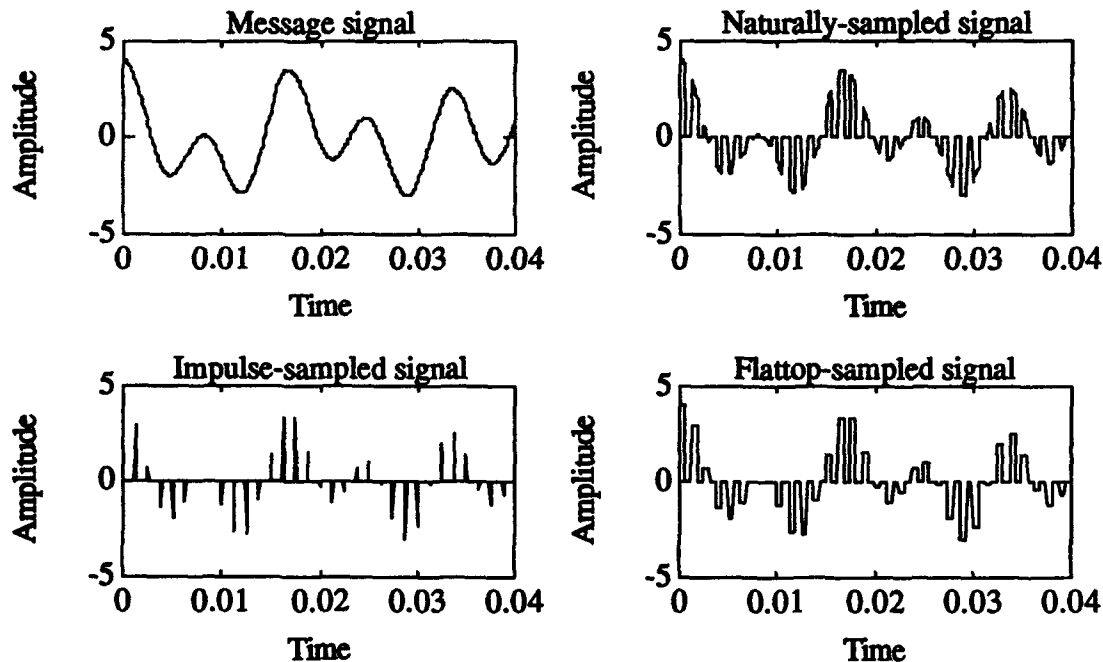
algorithm:

The sampling functions each begin by generating an impulse train. **impsamp** multiplies the impulse train and the input signal "s" and returns the product as "impsig."

flattop generates "impsig" in the same way, and then convolves it with a single pulse of duty cycle "d" to produce the flattop-sampled signal "flatsig."

natsamp creates a pulse train by convolving a single pulse of duty cycle "d" with the impulse train. The pulse train and the input signal "s" are multiplied to generate the naturally-sampled signal "natsig."

impsamp also returns its impulse train, and **flattop** and **natsamp** each return a pulse train.



User's Guide to the Communications Toolbox for MATLAB—page 11

fm_mod

function call:

`[fm_sig,delta_f,beta]=fm_mod(t,Ac,fc,fm,theta,delta_f,beta)`

synopsis:

`fm_mod` calculates either "delta_f" (the maximum frequency deviation) or "beta," whichever was not passed in, and returns both, along with the frequency-modulated signal "fm_sig." `fm_mod` will generate single- and multi-tone FM signals.

For multi-tone signals, frequency and phase parameters are represented by vectors. A sample vector for f_m , representing a three-tone signal, follows:

`fm = [50 150 230];`

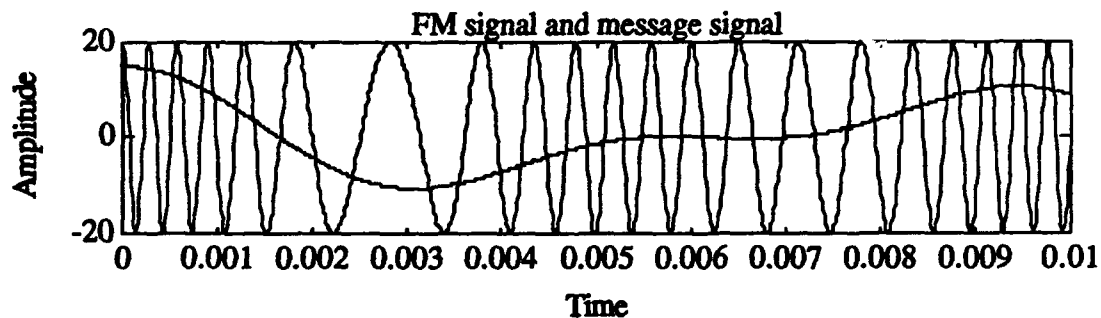
The vector representing phases for the multi-tone signal is established in the same way (even when all values are zero). The vectors representing frequency and phase must be the same length.

algorithm:

`fm_mod` is based on the formula

$$A_c \cos \left[2\pi f_c t + \beta \sum_i \sin(2\pi f_{m_i} t + \theta_i) \right]$$

where A_c is the amplitude of the FM signal, f_c is the frequency of the FM signal, f_m is the frequency of the message signal (a vector for multi-tone FM signals), and θ is the phase of the message signal (a vector for multi-tone FM signals) (Haykin, p. 190).



freq_div

function call:

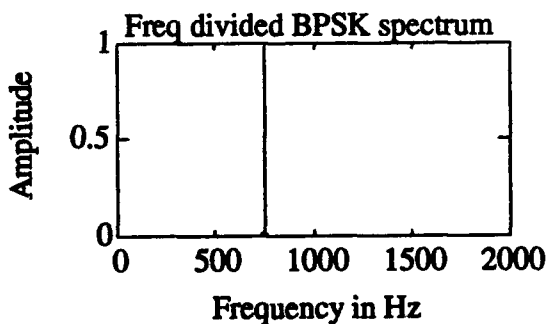
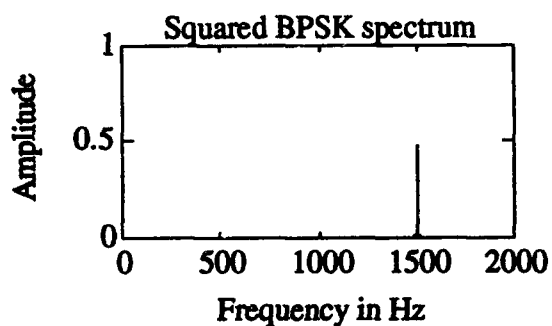
`div_sig=freq_div(squarebpsk,t,fc)`

synopsis:

`freq_div` accepts the squared BPSK signal "squarebpsk" and performs a frequency division in preparation for coherent detection. `freq_div` is specific to the BPSK detection process used in this laboratory set.

algorithm:

`freq_div` divides the squared BPSK signal by $\cos(2\pi f_c t)$, simulating its frequency shift down to the value of " f_c ."



fsk

function call:

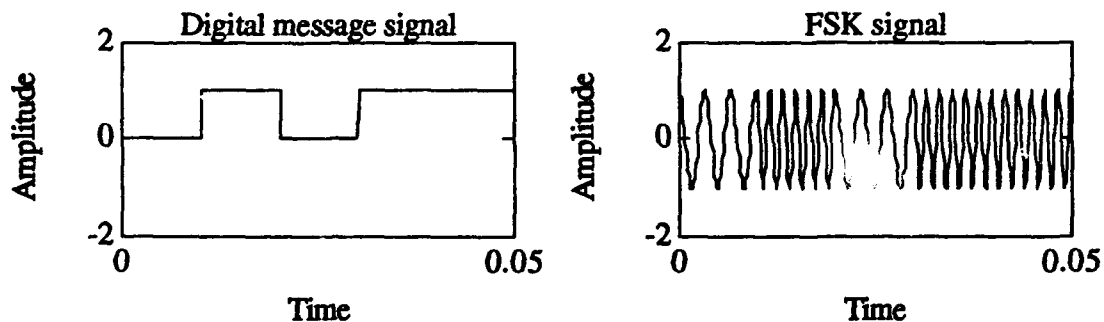
```
fsksig=fsk(dig_sig,delta_t,bitrate,freq_0,freq_1)
```

synopsis:

fsk performs frequency-shift-keying on the input digital signal "dig_sig", returning "fsksig," with "freq_0" assigned to 0 bits, and "freq_1" assigned to 1 bits.

algorithm:

fsk works bit by bit to assign one of two input frequency values to a frequency vector. The output FSK signal is generated as a cosine function of the frequency vector. "Delta_t" and "bitrate" are passed in to determine the number of points in each bit.



User's Guide to the Communications Toolbox for MATLAB—page 14

highpass, idealbnd, idealhi, ideallow, lowpass

function calls:

```
HPF=highpass(Hz,cutoff)
IBPF=idealbnd(Hz,cutoff1,cutoff2)
IHPF=idealhi(Hz,cutoff)
ILPF=ideallow(Hz,cutoff)
LPF=lowpass(Hz,cutoff)
```

synopsis:

These filters each return a vector the length of "Hz" (returned from `spectral`). Examining a one-sided spectral analysis graph ("specsig" plotted against "Hz") will help in determining the desired cutoff frequency or frequencies.

The "feval" command in MATLAB allows the filter functions to be called from within the functions `recoverm` and `recovers`.

algorithm:

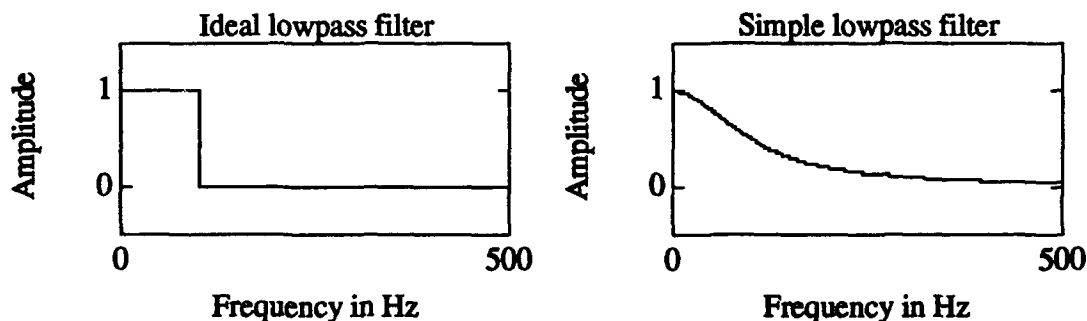
The simple lowpass filter in `lowpass` is based on the following formula (Stanley, p. 98):

$$H(f) = \frac{1}{1 + j \frac{f}{f_c}}$$

The simple highpass filter in `highpass` is based on the following formula:

$$H(f) = \frac{f}{1 + j \frac{f}{f_c}}$$

The remaining "ideal" filters are composed of 0's and 1's, changing values at the cutoff frequency or frequencies to retain only the desired frequency band. Each of the lowpass filters below has a cutoff frequency of 100 Hz.



manchest, nrzlb, nrzluni, rzuni

function calls:

```
[manchsig,bit_t,transec]=manchest(codedsig,delta_t,bitrate)
[nrzlsig,bit_t,transec]=nrzlb(codedsig,delta_t,bitrate)
[nrzlsig,bit_t,transec]=nrzluni(codedsig,delta_t,bitrate)
[rzumisig,bit_t,transec]=rzuni(codedsig,delta_t,bitrate)
```

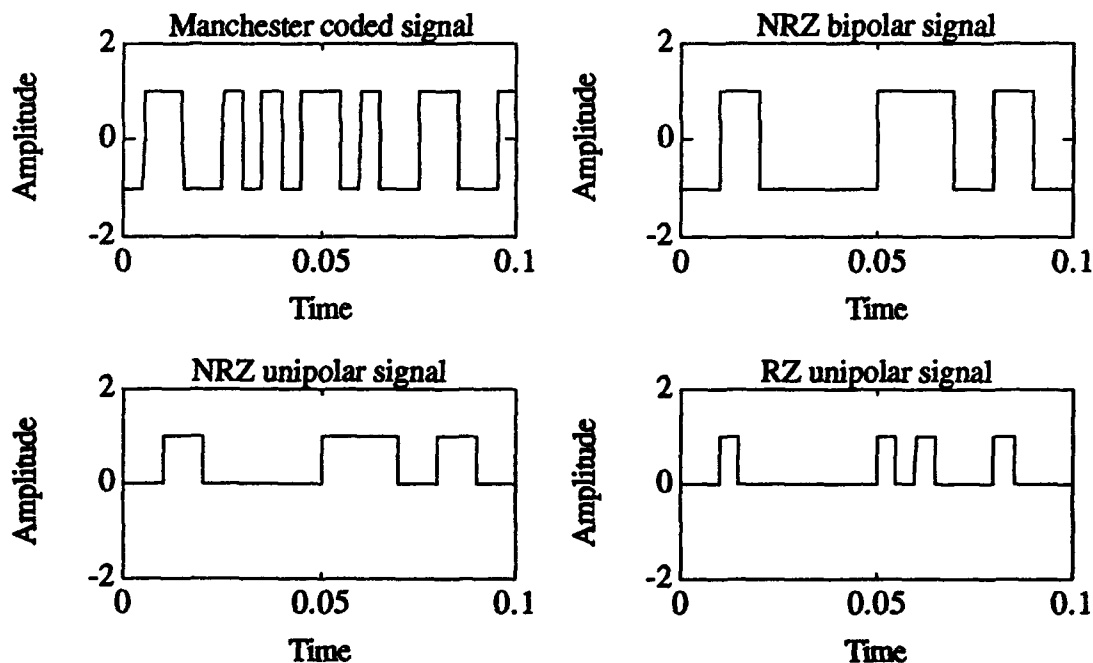
synopsis:

Each of these four digital encoding functions returns a digital signal at the specified bitrate for plotting against the time vector "bit_t." The number of transmission seconds "transec" is calculated.

manchest returns a manchester-encoded signal. **nrzlb** returns a bipolar non-return-to-zero signal. **nrzluni** returns a unipolar non-return-to-zero signal. **rzuni** returns a unipolar return-to-zero signal.

algorithm:

These functions require input of a bitstream of 1's and 0's, such as "codedsig," returned from **encode**, or one randomly generated using the "rand" command in MATLAB. A time vector is established and the number of points per bit is determined. Values within the bit are assigned according to the encoding scheme. The examples below show the output for each encoding function based on input vector of [0 1 0 0 1 1 0 1 0].



User's Guide to the Communications Toolbox for MATLAB—page 16

par_ser, ser_par

function calls:

```
comb_sig=par_ser(odd_sig,even_sig,delta_t,bitrate)
[odd_sig,even_sig]=ser_par(dig_sig,delta_t,bitrate)
```

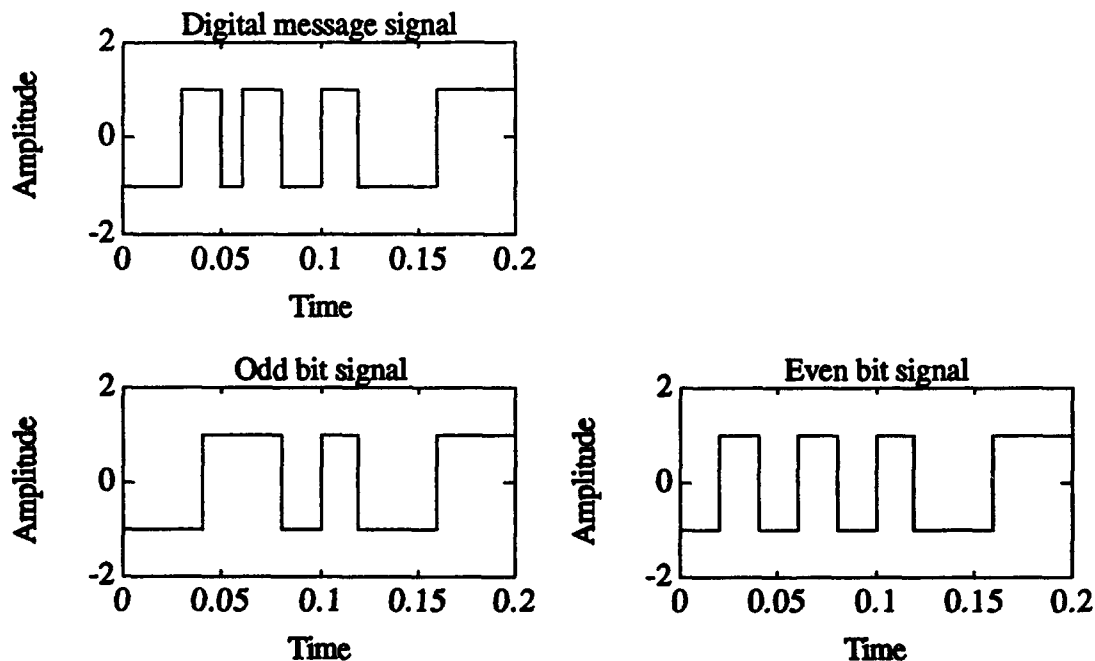
synopsis:

ser_par and **par_ser** contribute to the quadriphase-shift-keying (QPSK) process. **ser_par** is used during signal generation to split the digital signal into two signals, one composed of odd bits and one of even bits. The bitrates of the output signals are slowed to half that of the input signal. **par_ser** is used during the coherent detection of the QPSK signal to join the two signals into one by alternating the retrieval of the bits between the odd and even signals and reverting to the original bitrate.

algorithm:

ser_par presents some indexing challenges, since the number of points in successive bits is subject to change. The number of points in the first two bits of the input digital signal are applied to the first bit in the odd signal and the first bit in the even signal, in order to reduce the bitrate in half, while applying the correct bit values.

In **par_ser**, only the first half of each 'odd' bit is fetched, and the second half of each 'even' bit. This procedure controls the changing number of points per bit, and increases the bitrate to twice that of the input signals.



User's Guide to the Communications Toolbox for MATLAB—page 17

psd, spectral

function calls:

```
[psdsig,Hz,fftsig]=psd(anysig,delta_t)
[specsig,Hz,fftsig]=spectral(anysig,delta_t)
```

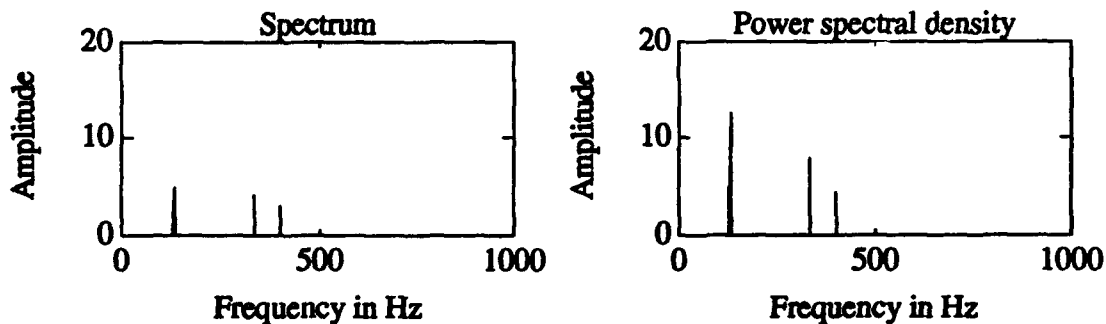
synopsis:

spectral and **psd** return vectors used for plotting one-sided spectral analysis graphs. Plot "specsig" against "Hz" to observe the amplitude of signal tones. Plot "psdsig" against "Hz" to observe the power present in signal tone (power spectral density).

To find average signal power, integrate "psdsig" by using the MATLAB "sum" command. The returned vector "fftsig" is used in signal recovery.

algorithm:

Each function performs a fast fourier transform on "anysig," creating the "fftsig" output signal, and then finds its absolute value. The first half of the vector is retained and divided by its length, resulting in "specsig," returned from **spectral**. In **psd**, the vector is then squared and divided by 2 to reflect the power in each signal tone, and returned as "psdsig." The "Hz" vector is scaled to the length of the Nyquist frequency ($1/2 \cdot \text{delta_t}$).



pulspos, pulswid

function calls:

```
ppsig=pulspos(s,delta_t,samprate,pulsdur)
pwsig=pulswid(s,delta_t,samprate,maxdur)
```

synopsis:

pulspos returns a pulse-position-modulated signal with pulse amplitude set at half of the maximum amplitude of the input signal. Offset values are calculated from the beginning of the sampling period T .

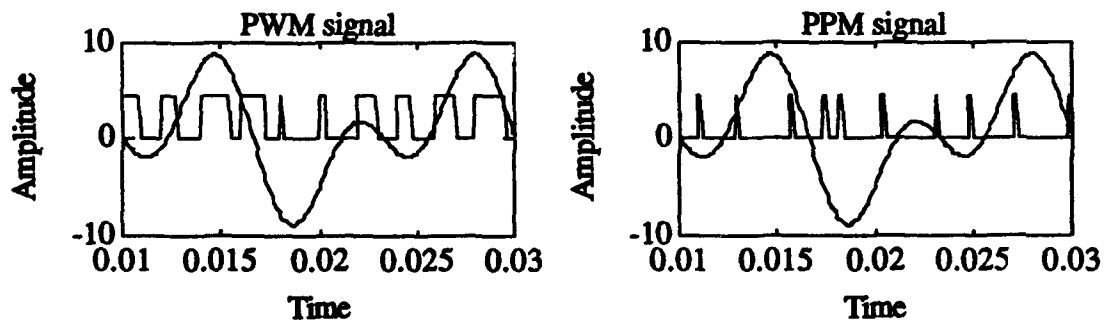
pulswid returns a pulse-width-modulated signal with pulse amplitude set at half of the maximum amplitude of the input signal. The input parameter "maxdur" sets the maximum pulse duration. All pulses are non-zero-width pulses.

algorithm:

The average number of points in the sampling period T and the amplitude of the pulse-modulated signal is established. The input signal is raised to all-positive values.

In **pulswid**, the signal amplitude at the pulse beginning is compared to the maximum signal amplitude. The ratio is applied to the number of points in "maxdur" to determine the number of points in the pulse duration "tau."

In **pulspos**, "tau" is fixed based on "pulsdur." The number of offset points from the pulse beginning is determined for each sampling period T , based on the ratio of the signal amplitude to the maximum signal amplitude at the pulse beginning.



quantize, quantuni

function calls:

```
[quanch_x,quanch_y,quan_sig,bin_nums]=quantize(symbols,elements,ss,samprate,delta_t)  
[quanch_x,quanch_y,quan_sig,bin_nums]=quantuni(symbols,elements,ss,samprate,delta_t)
```

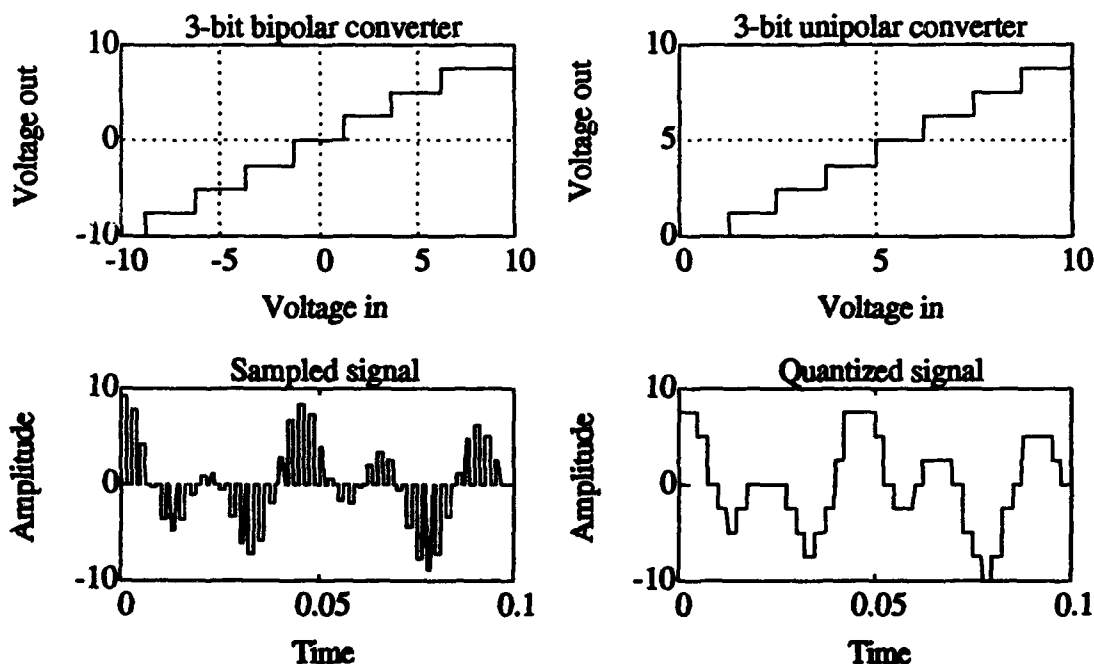
synopsis:

quantize and **quantuni** simulate analog-to-digital converters while providing two vectors, "quanch_x" and "quanch_y," for plotting the desired quantization characteristic (use the MATLAB "stairs" command). **quantize** simulates a bipolar quantizer with a range of -10 to +10 volts. **quantuni** simulates a unipolar quantizer with a range of 0 to +10 volts..

algorithm:

quantize and **quantuni** compute the number of converter levels by raising "symbols" to "elements." The value of each step in the converter is found by dividing the number of levels into the difference between the minimum and maximum converter values. The value of each successive level is found by adding one step, and is stored in the vector "quanch_x" (voltage in). "Quanch_y" stores output values associated with input values (voltage out).

The average number of points in the sampling period T is determined. The input signal value at the beginning of each sampling period is found and the value is assigned to the proper "bin." Intermediate values are rounded in **quantize** and truncated in **quantuni**. The bin numbers are stored in "bin_nums."



User's Guide to the Communications Toolbox for MATLAB—page 20

recoverm, recovers

function calls:

```
recsig=recoverm(fftsig,func,H,cutoff1,cutoff2)
recsig=recovers(fftsig,d,func,H,cutoff1,cutoff2)
```

synopsis:

recoverm and **recovers** filter and recover signals, returning the time-domain signal "recsig." Use **recoverm** for modulated signals and **recovers** for sampled signals. The input parameters "fftsig" (the fast fourier transform) and "H" are returned from **spectral**. The input parameter "func" refers to one of the filter functions included in the Communications Toolbox: **idealhigh**, **ideallow**, **idealbnd**, **lowpass**, and **highpass**. The name of the filter function must be enclosed by single quotes when passed in as a parameter.

algorithm:

The filter function passed in as "func" is evaluated using the cutoff frequency or frequencies, returning a filter the length of "H." The filter is tailored to the length of "fftsig" and the two vectors are multiplied.

In **recovers**, the resulting signal vector is multiplied by $1/d$ (the duty cycle) to scale the amplitude properly. Impulse-sampled signals require the input parameter "d" to be calculated as the sampling rate times Δt (the step size in the time vector).

Finally, an inverse fast fourier transform is performed on the signal vector.

snr

function call:

`s_n_ratio=snr(cleansig,noisysig)`

synopsis:

snr returns the signal-to-noise ratio in dB for the "noisysig" signal input.

Written by Mike Shields 28 Jul 93

algorithm:

snr is based on the following formula

$$10\log\left(\frac{\sum x^2}{\sum y^2}\right)$$

where "x" is the input "clean" signal, and "y" is the output, or "noisy," signal.

Index

compress	7
convert	8
conv_am	9
encode	8
envelope	10
expand	7
flattop	11
fm_mod	12
freq_div	13
fsk	14
highpass	15
hilbert	9
idealbnd	15
idealhi	15
ideallow	15
impsamp	11
lowpass	15
manchester	16
natsamp	11
nrzlb	16
nrzluni	16
par_ser	17
psd	18
pulsp	19
pulswid	19
quantize	20
quantuni	20
recoverm	21
recovers	21
rzuni	16
ser_par	17
snr	22
spectral	18
ssb	9

**This page is intentionally
left blank.**

APPENDIX G—COMPARISON OF REQUIREMENTS

Comparison of Laboratory Requirements

	Questions	Labeled Plots	Combined	% of Total
Comp-Aid 1	2	3	5	4%
Comp-Aid 2	8	7	15	12%
Comp-Aid 3	4	5	9	7%
Comp-Aid 4	9	3	12	10%
Comp-Aid 5	4	6	10	8%
Comp-Aid 6	3	6	9	7%
Comp-Aid 7	6	5	11	9%
Comp-Aid 8	13	12	25	21%
Comp-Aid 9	9	16	25	21%
Total	58	63	121	100%

	Questions	Labeled Plots	Plots	M-file Ops	Combined	% of Total
Prog 1	2	3	6	8	19	4%
Prog 2A	8	3	8	10	29	6%
Prog 2B	8	3	8	10	29	6%
Prog 2C	6	2	6	8	22	5%
Prog 3A	4	2	4	6	16	4%
Prog 3B	4	2	4	6	16	4%
Prog 4A	5	0	6	10	21	5%
Prog 4B	8	0	7	10	25	6%
Prog 4C	4	0	7	14	25	6%
Prog 5	4	6	14	18	42	9%
Prog 6	4	6	22	31	63	14%
Prog 7	4	4	13	22	43	10%
Prog 8	9	8	12	19	48	11%
Prog 9	4	14	16	17	51	11%
Total	74	53	133	189	449	100%

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2. Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3. Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
4. Randy L. Borchardt, Code EC/Bt Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	2
5. Dr. Dan C. Boger, Code AS/Bo Department of Systems Management Naval Postgraduate School Monterey, CA 93943-5103	1
6. LCDR Michael K. Shields, USN, Code EC/SI Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
7. Capt Michael A. Cervi, USAF, Code CC/Ce Joint C3 Academic Group Naval Postgraduate School Monterey, CA 93943-5142	1
8. Computer Technology Programs Code 37 Naval Postgraduate School Monterey, CA 93943-5119	1